第五章 二维变换与裁剪

计算机图形学

建模、变换、绘制和显示

显示

完成三维物体建模和绘制后，还要在显示器平面上将其呈现出来。首先借助投影变换绘制投影图，其次在图像显示器等光栅输出设备是图形用像素的集合来表示。（第3、4章）

流水线：建模、变换、绘制和显示

- 变换

对景物几何模型施加某种处理，把处理前的景物形状数据转换为处理后的景物形状数据的数学过程。如旋转、移动、缩放等。（第5、6章）

本章主要内容：

- 5.1 图形几何变换基础
- 5.2 二维图形的基本几何变换矩阵
- 5.3 二维图形的复合变换
- 5.4 二维图形裁剪
- 5.5 Cohen-Sutherland直线段裁剪算法
- 5.6 中点分割直线段裁剪算法
- 5.7 Liang-Barsky直线段裁剪算法
- 5.8 多边形裁剪算法
- 5.9 本章小结
- 习题5

本章主要内容：

- 5.1 图形几何变换基础
- 5.2 二维图形的基本几何变换矩阵
- 5.3 二维图形的复合变换
- 5.4 二维图形裁剪
- 5.5 Cohen-Sutherland直线段裁剪算法
- 5.6 中点分割直线段裁剪算法
- 5.7 Liang-Barsky直线段裁剪算法
- 5.8 多边形裁剪算法
- 5.9 本章小结
- 习题5

本章学习目标：

- 了解齐次坐标的概念
- 掌握掌握二维基本几何变换矩阵
- 熟练掌握Cohen-Sutherland直线段裁剪算法
- 掌握中点分割直线段裁剪算法
- 掌握Liang-Barsky直线段裁剪算法
- 了解Sutherland-Hodgman多边形裁剪算法

矩阵乘法

- 由线性代数知道，矩阵乘法不满足交换律，只有左矩阵的列数等于右矩阵的行数时，两个矩阵才可以相乘。特别地，对于二维变换的两个3×3的方阵A和B，矩阵相乘公式为：

$$A \cdot B = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \cdot \begin{bmatrix} b_{11} & b_{12} & b_{13} \\ b_{21} & b_{22} & b_{23} \\ b_{31} & b_{32} & b_{33} \end{bmatrix}$$

$$= \begin{bmatrix} a_{11}b_{11}+a_{12}b_{21}+a_{13}b_{31} & a_{11}b_{12}+a_{12}b_{22}+a_{13}b_{32} & a_{11}b_{13}+a_{12}b_{23}+a_{13}b_{33} \\ a_{21}b_{11}+a_{22}b_{21}+a_{23}b_{31} & a_{21}b_{12}+a_{22}b_{22}+a_{23}b_{32} & a_{21}b_{13}+a_{22}b_{23}+a_{23}b_{33} \\ a_{31}b_{11}+a_{32}b_{21}+a_{33}b_{31} & a_{31}b_{12}+a_{32}b_{22}+a_{33}b_{32} & a_{31}b_{13}+a_{32}b_{23}+a_{33}b_{33} \end{bmatrix}$$

5.1 图形几何变换基础

- 在计算机图形学中，通常需要将画出的图形平移到某一位置，或改变图形的大小和形状，或利用已有图形生成复杂图形，这种图形处理的过程就是图形的几何变换，简称图形变换。
- 通过对图形进行几何变换，可以由简单图形构造复杂图形。

5.1 图形几何变换基础

- 二维图形和三维图形都可以进行图形变换，二维图形几何变换是三维图形几何变换的基础
- 图形变换通常采用矩阵的方法，图形所做的变换不同其变换矩阵也不同。变换的实质是对由图形上各点的坐标组成的矩阵进行运算，因此在讨论各种具体图形几何变换时，可以归结为一个点的变换。
- 图形的几何变换表达为图形顶点几何矩阵与某一变换矩阵相乘的问题，引入规范化齐次坐标。

5.1.1 规范化齐次坐标

- 用n+1维的向量表示n维向量,即用高维数据表示低维数据

在二维平面中，点$P(x,y)$的齐次坐标表示为(wx,wy,w)。如齐次坐标（8,4,2）和（4,2,1）都是表示点(2,1)。

在三维空间中，点$P(x,y,z)$的齐次坐标表示为(wx,wy,wz,w)。

当$w=1$时，称为规范化齐次坐标。二维点$P(x,y)$的规范化齐次坐标表示为$(x,y,1)$

定义了规范化齐次坐标后，图形的几何变换可以表达为图形顶点集合的规范化齐次坐标矩阵与某一变换矩阵相乘的形式。

5.1.1 规范化齐次坐标

- 《计算机图形学(OpenGL版)》的作者F.S. Hill Jr.曾说过一句话：

"齐次坐标表示是计算机图形学的重要手段之一，它既能够用来明确区分向量和点，同时也更易用于进行仿射（线性）几何变换。"

- 因为该坐标允许平移、旋转、比例以及投影等可表示为矩阵与向量相乘的一般向量运算。
- 依据链式法则，任何此类运算的序列均可相乘为单一个矩阵，从而实现简单且有效之处理。与此相反，若使用笛卡儿坐标，平移及透视投影不能表示成矩阵相乘，虽然其他的运算可以。

5.1.1 规范化齐次坐标

■《计算机图形学(OpenGL版)》的作者F.S. Hill Jr.曾说过一句话：

"齐次坐标表示是计算机图形学的重要手段之一，它既能够用来明确区分向量和点，同时也更易用于进行仿射（线性）几何变换。"

- 表示点：$(x, y, w = 1)$
- 表示向量：$(x, y, w = 0)$

5.1.2 矩阵相乘

- 二维图形顶点表示为规范化齐次坐标后，其图形顶点集合矩阵一般为n*3的矩阵，其中n为顶点数，变换矩阵为3*3的矩阵。在进行图形几何变换时需要用到线性代数里的矩阵相乘运算。例如，对于n*3的矩阵P和3*3的矩阵T，令m=n-1,则矩阵相乘的公式为

$$P \bullet T = \begin{bmatrix} p_{00} & p_{01} & p_{02} \\ p_{10} & p_{11} & p_{12} \\ \mathrm{M} & & \\ p_{m0} & p_{m1} & p_{m2} \end{bmatrix} \bullet \begin{bmatrix} t_{00} & t_{01} & t_{02} \\ t_{10} & t_{11} & t_{12} \\ t_{20} & t_{21} & t_{22} \end{bmatrix}$$

5.1.3 二维图形的几何变换

- 用规范化齐次坐标表示的二维图形几何变换矩阵是一个3*3的方阵，简称二维几何变换矩阵，形式如下

$$T=\begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

矩阵的每一个元素都有特殊含义

5.1.3 二维图形的几何变换

- 将二维几何变换矩阵的分为4个子矩阵

$$T_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$$

对图形进行比例、旋转、反射和错切变换

$$T_2 = \begin{bmatrix} l & m \end{bmatrix}$$

对图形进行平移变换

5.1.3 二维图形的几何变换

- 将二维几何变换矩阵的分为4个子矩阵

$$T_3 = \begin{bmatrix} p \\ q \end{bmatrix}$$

对图形进行投影变换

$$T_4 = [s]$$

对图形整体进行比例变换

5.1.4 物体变换与坐标变换

- 同一种变换可以看作是物体变换，也可以看作是坐标变换。
  - 物体变换： 使用同一变换矩阵作用于物体的所有顶点，但坐标系位置不发生改变。
  - 坐标变换：坐标系发生变换，但物体的位置不发生改变，然后在新坐标系下表示所有物体的顶点。

5.1.5二维几何变换形式

- 二维几何变换的基本方法是把变换矩阵作为一个算子，作用到变换前的图形顶点集合的规范化齐次坐标矩阵上，得到变换后新的图形顶点集合的规范化齐次坐标矩阵。连接变换后的新图形顶点，就可以绘制出变换后的二维图形。

变换前图形顶点集合的规范化齐次坐标矩阵为：

$$P=\begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdots & \cdots & \cdots \\ x_n & y_n & 1 \end{bmatrix}$$

变换后图形顶点集合的规范化齐次坐标矩阵为：

$$P'=\begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ \cdots & \cdots & \cdots \\ x'_n & y'_n & 1 \end{bmatrix}$$

5.1.3 物体变换与坐标变换

- 则二维几何变换公式为，可以写成：

$$P' = P \cdot T$$

$$\begin{bmatrix} x'_1 & y'_1 & 1 \\ x'_2 & y'_2 & 1 \\ \cdots & \cdots & \cdots \\ x'_n & y'_n & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ \cdots & \cdots & \cdots \\ x_n & y_n & 1 \end{bmatrix} \cdot \begin{bmatrix} a & b & p \\ c & d & q \\ l & m & s \end{bmatrix}$$

变换后图形顶点集合的规范化齐次坐标阵 $N*3$

变换前图形顶点集合的规范化齐次坐标矩 $N*3$

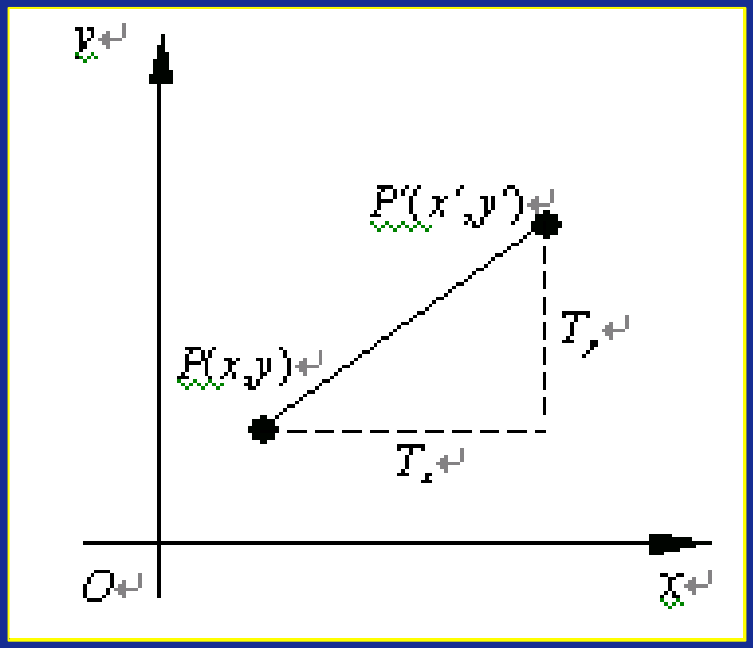
变换矩阵 $3*3$

5.2 二维图形基本几何变换矩阵

- 二维图形基本几何变换是指相对于坐标原点和坐标轴进行的几何变换
- 包括平移、比例、旋转、反射和错切五种变换。

平移变换

- 平移转换是指将$P(x,y)$点移动到$P'(x,y)$位置的过程，即物体从一个位置到另一位置所作的直线移动。



- 如果要把一个位于的点移到新位置时，只要在原坐标上加上平移距离T_x及T_y即可

平移变换

- 坐标表示

$$\begin{cases} x' = x + T_x \\ y' = y + T_y \end{cases}$$

相应的齐次坐标矩阵表示为

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x + T_x & y + T_y & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \bullet \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

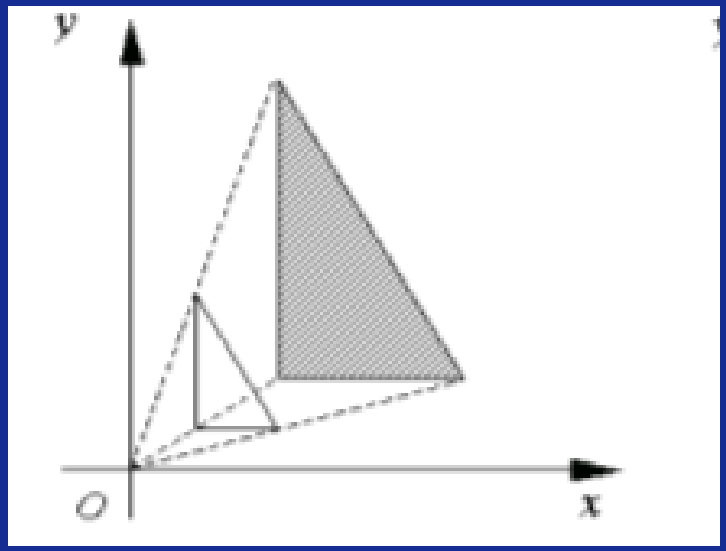
因此，二维平移变换矩阵

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ T_x & T_y & 1 \end{bmatrix}$$

其中，T_x T_y为平移参数

比例变换

- 比例变换是指将$P(x,y)$点相对于原点O，沿x方向缩放S_x倍，y方向缩放S_y倍，得到$P'(x,y)$点位置的过程。



比例变换

- 坐标表示

$$\begin{cases} x' = S_x \cdot x \\ y' = S_y \cdot y \end{cases}$$

相应的齐次坐标矩阵表示为

$$\begin{bmatrix} x' & y' & 1 \end{bmatrix} = \begin{bmatrix} x \cdot S_x & y \cdot S_y & 1 \end{bmatrix} = \begin{bmatrix} x & y & 1 \end{bmatrix} \bullet \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因此，二维缩放变换矩阵

$$T = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

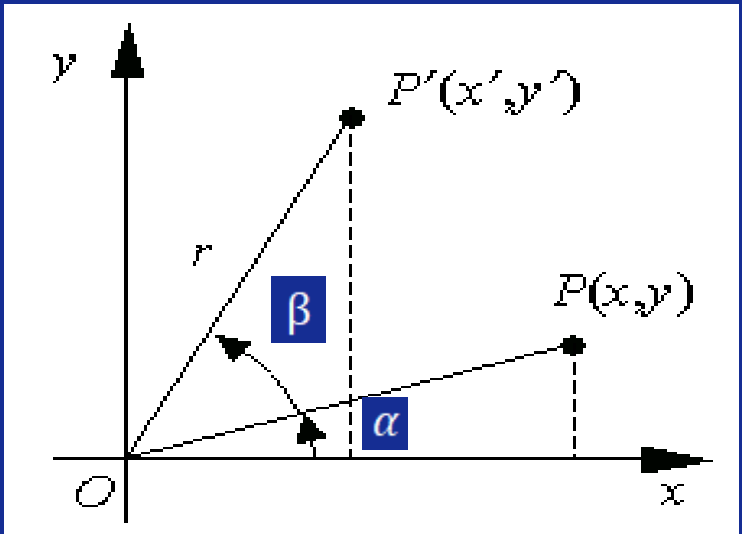
其中，S_x S_y为缩放参数

比例变换

- 比例变换可以改变图形的形状。
  - 当$S_x = S_y$且S_x、S_y大于1时，图形等比放大；
  - 当$S_x = S_y$且S_x、S_y小于1大于0时，图形等比缩小；
  - 当$S_x \neq S_y$时，图形发生形变。
- 前面介绍过变换矩阵的子矩阵 $T_4 = [s]$

旋转变换

- 旋转变换是指将$P(x,y)$点相对于原点O旋转一个角度θ（逆时针方向为正，顺时针方向为负），得到$P'(x,y)$点位置的过程。



$$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases}$$

旋转变换

- 坐标表示

$$\begin{cases} x' = r\cos(\alpha+\beta) = x\cos\beta - y\cos\beta \\ y' = r\sin(\alpha+\beta) = x\sin\beta - y\sin\beta \end{cases}$$

相应的齐次坐标矩阵表示为

$$[x' \quad y' \quad 1] = [x\cos\beta - y\sin\beta \quad x\sin\beta + y\cos\beta \quad 1] = [x \quad y \quad 1] \bullet \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

旋转变换

因此，二维旋转变换矩阵

$$T=\begin{bmatrix}\cos\beta & \sin\beta & 0\\ -\sin\beta & \cos\beta & 0\\ 0 & 0 & 1\end{bmatrix}$$

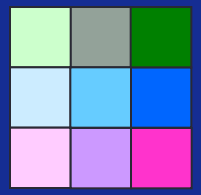
其中，α 为起始角，β 为旋转角

思考

- 请写出绕原点进行顺时针旋转的变换矩阵

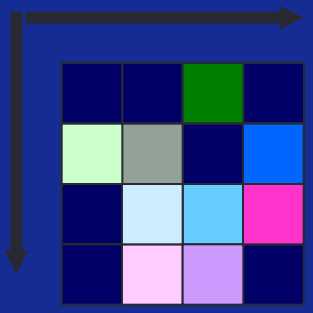
$$T=\begin{bmatrix}\cos(-\beta) & \sin(-\beta) & 0\\ -\sin(-\beta) & \cos(-\beta) & 0\\ 0 & 0 & 1\end{bmatrix}=\begin{bmatrix}\cos\beta & -\sin\beta & 0\\ \sin\beta & \cos\beta & 0\\ 0 & 0 & 1\end{bmatrix}$$

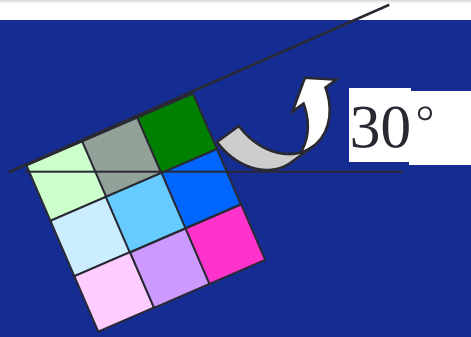
扩展题（研究生入学考试题）



$\theta = 30°$

$$\begin{cases} x' = 0.866x - 0.5y \\ y' = 0.5x + 0.866y \end{cases}$$





$x'_{min} = 0.866 - 0.5 * 3 = -0.634$

$x'_{max} = 0.866 * 3 - 0.5 = 2.098$

$y'_{min} = 0.866 + 0.5 = 1.366$

$y'_{max} = 0.866 * 3 + 0.5 * 3 = 4.098$

按照图像旋转计算公式获得的结果与想象中的差异很大。

$x:[1,3];\quad y:[1,3]$

$x':[-1,2];\quad y':[1,4]$

$x'':[1,4];\quad y'':[1,4]$

扩展题（研究生入学考试题）

- 旋转计算公式计算出的值为小数，而坐标值为正整数。
- 旋转计算公式计算的结果值所在范围与原来的值所在的范围不同。

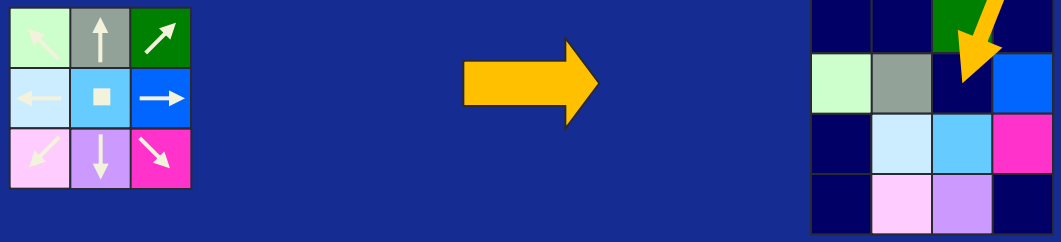
扩展题（研究生入学考试题）

- 图像旋转之后，出现了两个问题：

1）像素的排列不是完全按照原有的相邻关系。这是因为相邻像素之间只能有8个方向，如下图所示。

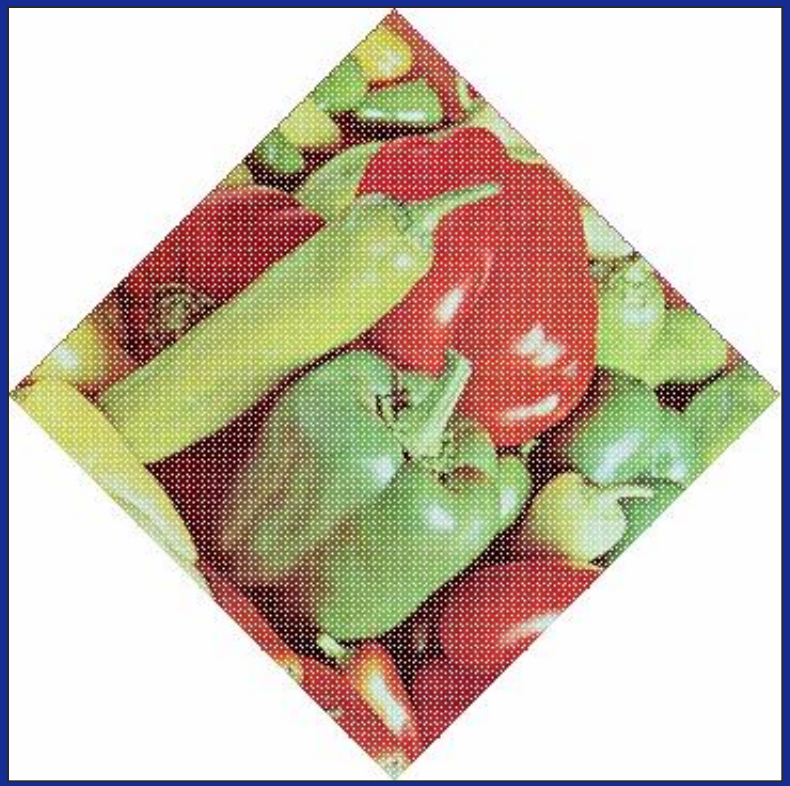
2）会出现许多的空洞点。

- 下面，我们通过一个实际例子，来看这两个问题带来的图像画面效果上的问题。



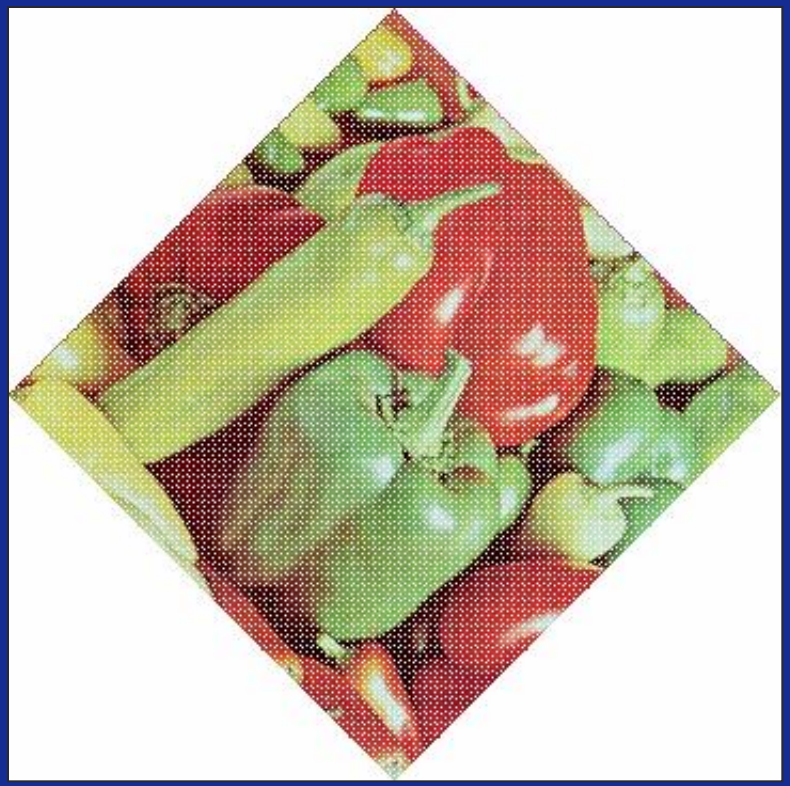
扩展题（研究生入学考试题）

- 图像的旋转结果



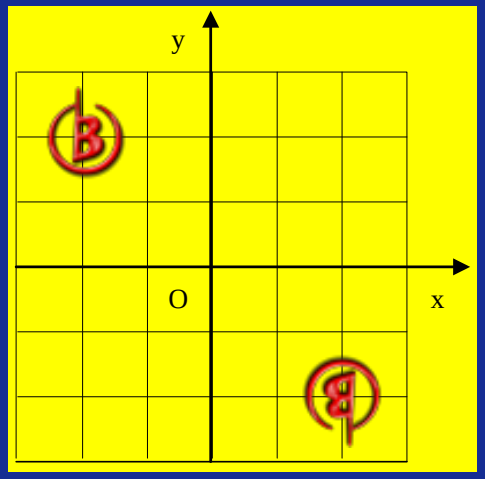
扩展题（研究生入学考试题）

- 图像旋转中的插值处理效果

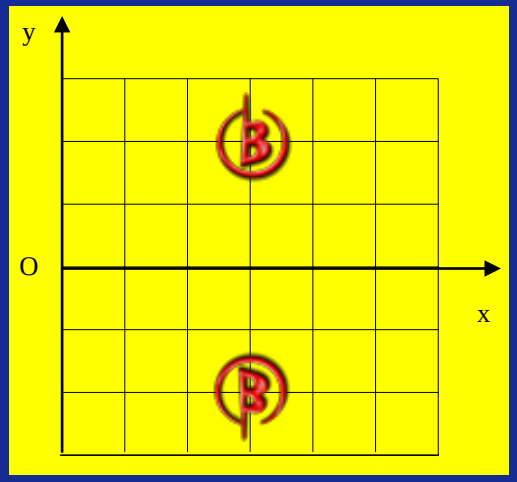


反射变换

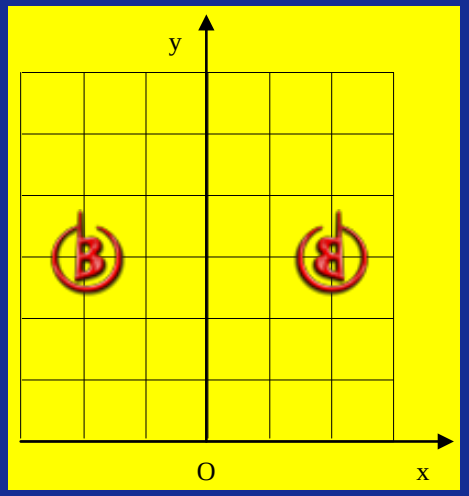
- 反射变换又称为对称变换是用来产生物体的镜象的一种变换。物体的镜象一般是相对于一对称轴生成的 。



关于坐标原点的反射变换



关于y轴的反射变换



关于x轴的反射变换

关于原点反射变换

- 坐标表示

$$\begin{cases} x' = -x \\ y' = -y \end{cases}$$

相应的齐次坐标矩阵表示为

$$[x' \quad y' \quad 1] = [-x \quad -y \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因此，关于原点二维反射变换矩阵

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

关于y轴反射变换

- 坐标表示

$$\begin{cases} x' = -x \\ y' = y \end{cases}$$

相应的齐次坐标矩阵表示为

$$[x' \quad y' \quad 1] = [-x \quad y \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因此，关于y轴的二维反射变换矩阵

$$T = \begin{bmatrix} -1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

关于x轴反射变换

- 坐标表示

$$\begin{cases} x' = x \\ y' = -y \end{cases}$$

相应的齐次坐标矩阵表示为

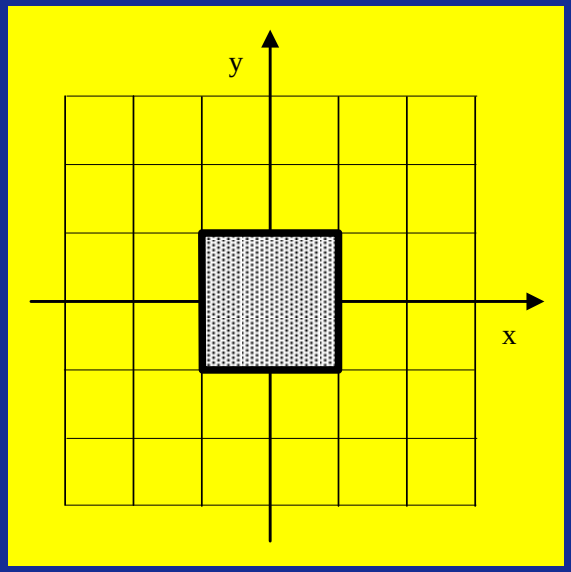
$$[x' \quad y' \quad 1] = [x \quad -y \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因此，关于x轴二维反射变换矩阵

$$T = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

错切变换

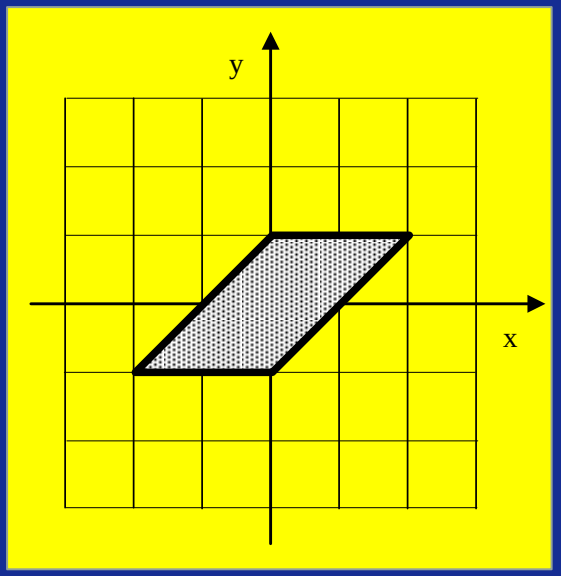
- 这种变换可使物体产生变形，即物体产生扭转或称为错切变换。具体操作是将坐标点沿x和y轴发生不等量的变换，得到点的过程。



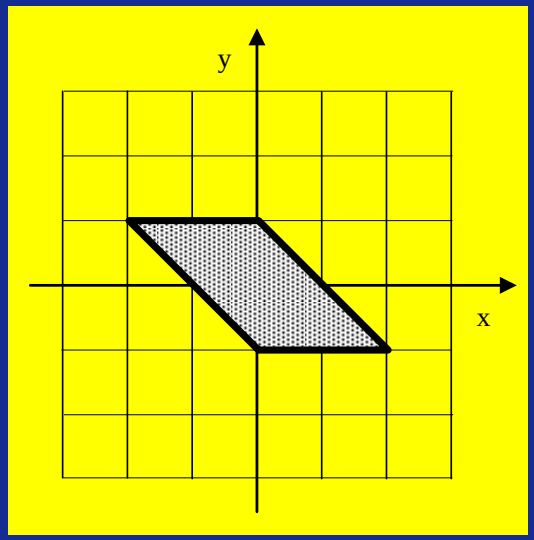
正方形

错切变换

- 沿x方向关于y轴的错切



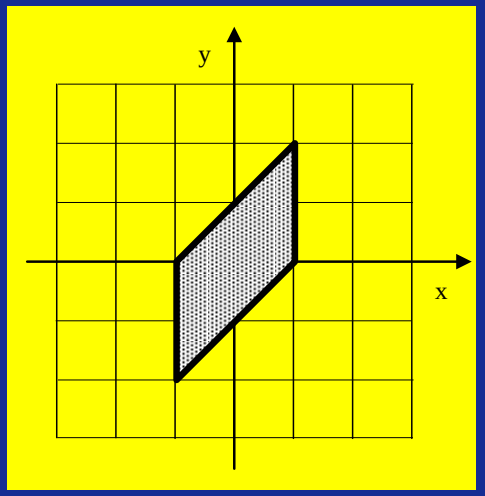
沿x+方向错切



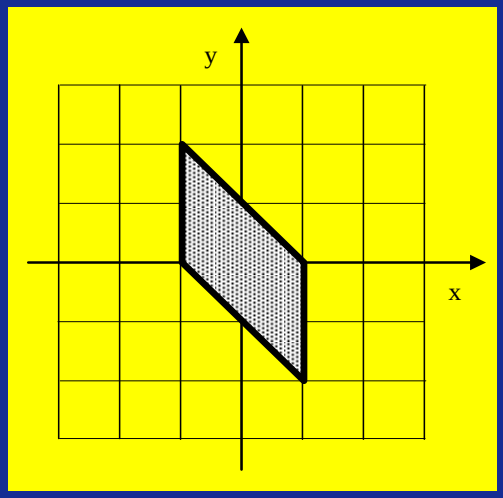
沿x-方向错切

错切变换

- 沿y方向关于x轴的错切



沿y+方向错切



沿y-方向错切

错切变换

错切变换

- 坐标表示

$$\begin{cases} x' = x + cy \\ y' = bx + y \end{cases}$$

相应的齐次坐标矩阵表示为

$$[x' \quad y' \quad 1] = [x + cy \quad bx + y \quad 1] = [x \quad y \quad 1] \cdot \begin{bmatrix} 1 & b & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

因此，沿x、y两个方向二维错切变换矩阵

$$T = \begin{bmatrix} 1 & b & 0 \\ c & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

错切变换

- 在前面的变换中，子矩阵 $T_1 = \begin{bmatrix} a & b \\ c & d \end{bmatrix}$

的非对角线元素大多为零，如果c和b不为零，则意味着对图形进行错切变换所示。

- 令b = 0可以得到沿x方向（y坐标不变）的错切变换，c>0是沿x正向的错切变换，c<0是沿x负向的错切变换。
- 令c = 0可以得到沿y方向（x坐标不变）的错切变换，b>0是沿y正向的错切变换，b<0是沿y负向的错切变换。

- 上面讨论的五种变换给出的都是点变换的公式，对于线框模型，图形的变换实际上都可以通过点变换来完成。
- 例如直线段的变换可以通过对两个顶点坐标进行变换，连接新顶点得到变换后的新直线；多边形的变换可以通过对每个顶点进行变换，连接新顶点得到变换后的新多边形来实现。曲线的变换可通过变换控制多边形的控制点并重新画线来完成。

二维仿射变换

- 符合下面形式的坐标变换称为二维仿射变换（Affine Transformation）

$$\begin{cases} x' = a_{11}x + a_{12}y + a_{13} \\ y' = a_{21}x + a_{22}y + a_{23} \end{cases}$$

- 仿射变换具有平行线变换成平行线，有限点映射到有限点的一般特性。平移、比例、旋转、反射和错切五种变换都是二维仿射变换的特例，任何一组二维仿射变换总可表示为这五种变换的组合。因此，平移、比例、旋转、反射的仿射变换保持变换前后两直线间的角度、平行关系和长度之比不改变。

5.3 二维图形复合变换

- 任意一个变换序列均可表示为一个组合变换矩阵。组合变换矩阵可由基本变换矩阵的乘积求得。由若干基本变换矩阵相乘求得复合变换矩阵的方法称为矩阵的级联。

$$P' = P \cdot T = P \cdot T_0 \cdot T_1 \cdot \cdots \cdot T_{n-1}$$

其中，T为复合变换矩阵，$T_0 \cdot T_1 \cdot \cdots \cdot T_{n-1}$为n个基本几何变换矩阵。

注意：注意矩阵的乘法顺序，不满足交换律，因此 $T_0 \cdot T_1 \neq T_1 \cdot T_0$

通常先计算 $T = T_0 \cdot T_1 \cdot \cdots \cdot T_{n-1}$，再计算 $P' = P \cdot T$

复合平移

- 对一物体连续平移两次，假定两次平移的距离为（x_1，y_1）及（x_2，y_2），则变换后的坐标为

$$P' = P \cdot T_1 \cdot T_2 = P \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 & y_1 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_2 & y_2 & 1 \end{bmatrix}$$

$$= P \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_1 + x_2 & y_1 + y_2 & 1 \end{bmatrix}$$

进行连续两次平移，实际上是把平移距离相加

复合缩放

- 对一物体连续缩放两次

$$P' = P \cdot T_1 \cdot T_2 = P \cdot \begin{bmatrix} S_{x1} & 0 & 0 \\ 0 & S_{y1} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} S_{x2} & 0 & 0 \\ 0 & S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= P \cdot \begin{bmatrix} S_{x1} \cdot S_{x2} & 0 & 0 \\ 0 & S_{y1} \cdot S_{y2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

进行连续两次平移，实际上是把相应的比例因子相乘

复合旋转

- 对一物体连续旋转两次

$$P' = P \mathrm{g} T_1 \mathrm{g} T_2 = P \mathrm{g} \begin{bmatrix} \cos\alpha & \sin\alpha & 0 \\ -\sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} \mathrm{g} \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

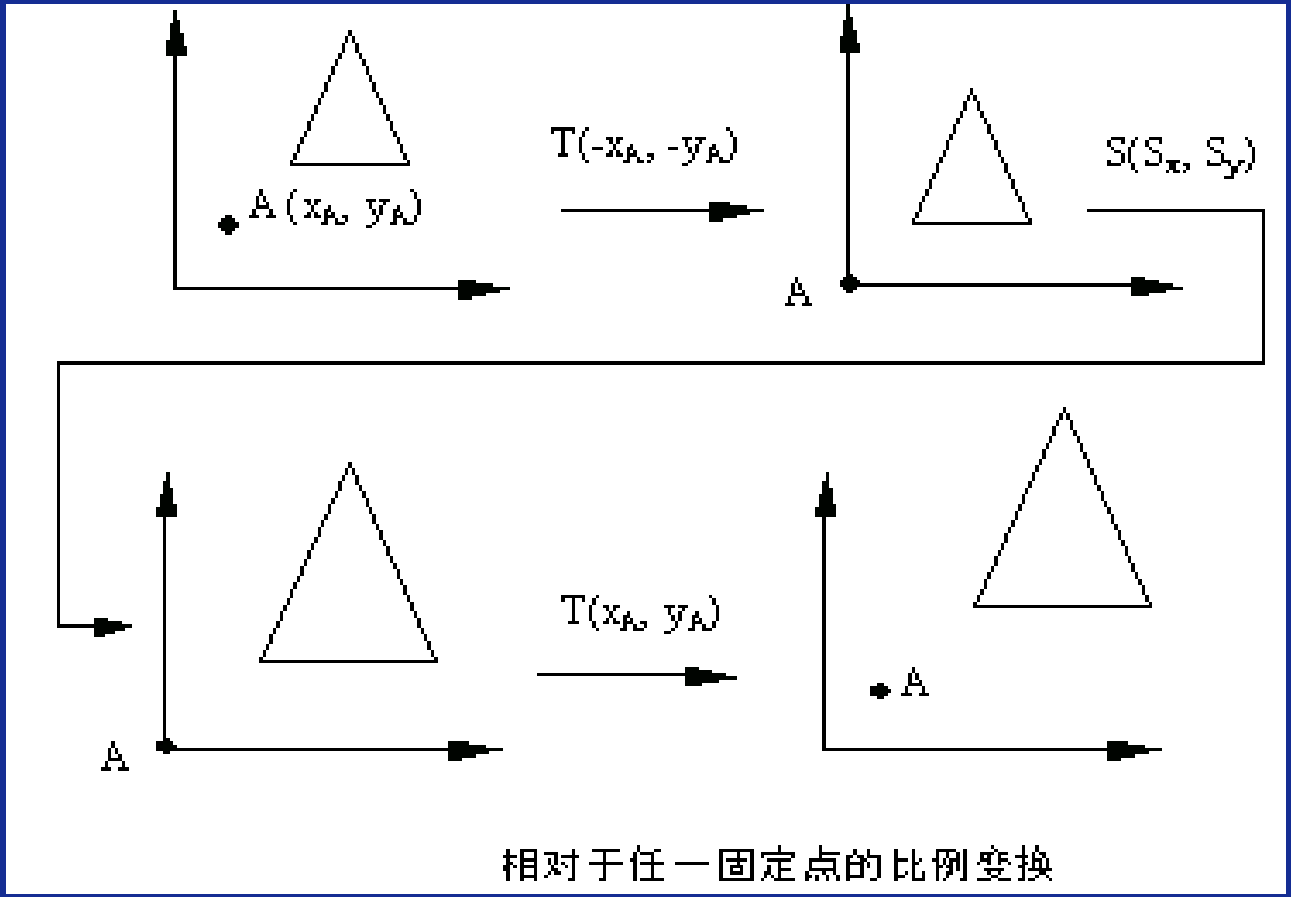
$$= P \mathrm{g} \begin{bmatrix} \cos\alpha\cos\beta - \sin\alpha\sin\beta & \cos\alpha\sin\beta + \sin\alpha\cos\beta & 0 \\ -\sin\alpha\cos\beta - \sin\alpha\sin\beta & -\sin\alpha\sin\beta - \cos\alpha\sin\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$= P \mathrm{g} \begin{bmatrix} \cos(\alpha+\beta) & \sin(\alpha+\beta) & 0 \\ -\sin(\alpha+\beta) & \cos(\alpha+\beta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

进行连续两次平移，实际上是把相应的旋转角度相加

5.3.2 相对于任意参考点的二维比例变换

- 相对于任一固定点$A(x_A, y_A)$的比例（缩放）变换



5.3.2 相对于任意参考点的二维比例变换

- (1)先将A点移到坐标原点

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix}$$

- (2)将物体进行缩放

$$T_2 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- (3)将A点移回到原位

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix}$$

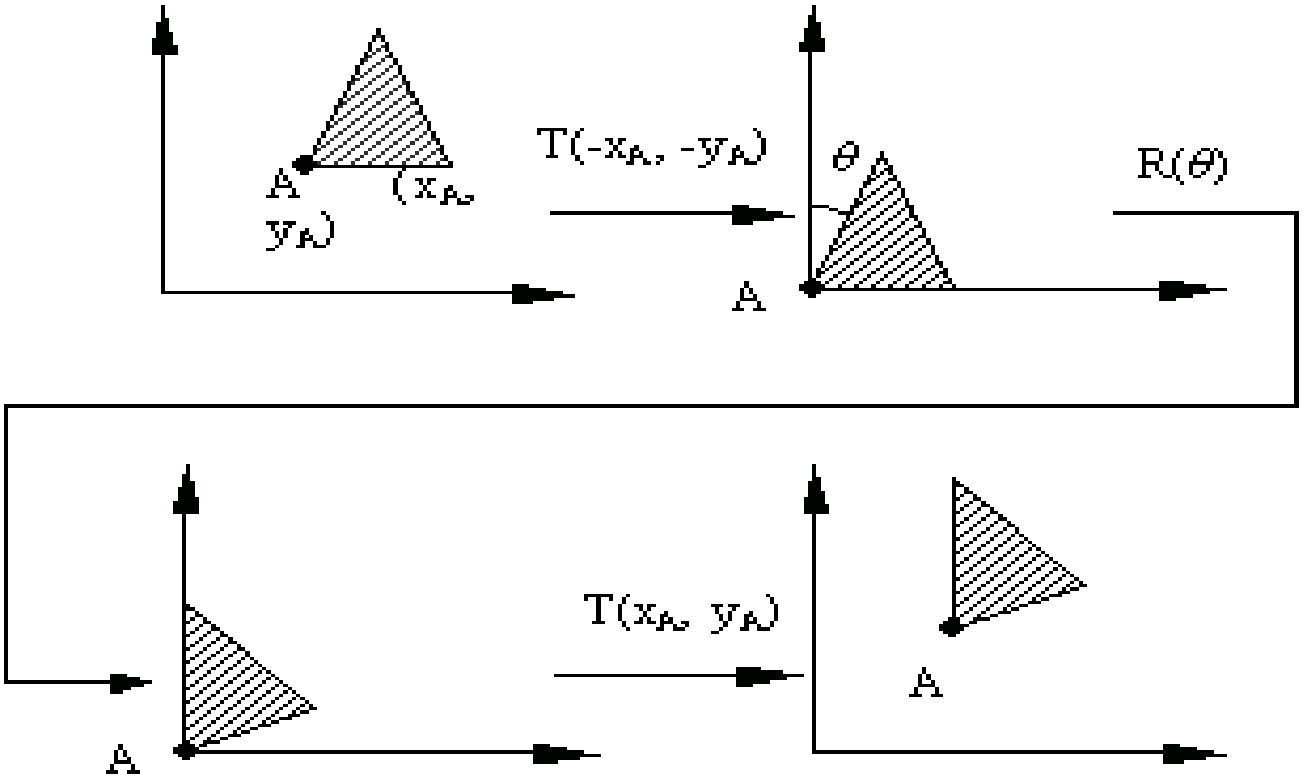
- 变换后的坐标

$$P' = P \cdot T_1 \cdot T_2 \cdot T_2$$

$$= P \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix} \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix}$$

5.3.2 相对于任意参考点的二维旋转变换

- 围绕任一固定点$A(x_A, y_A)$的旋转变换



围绕任一基准点的旋转变换

5.3.2 相对于任意参考点的二维旋转变换

- (1)先将A点移到坐标原点

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -x_A & -y_A & 1 \end{bmatrix}$$

- (2)将物体逆时针旋转 θ

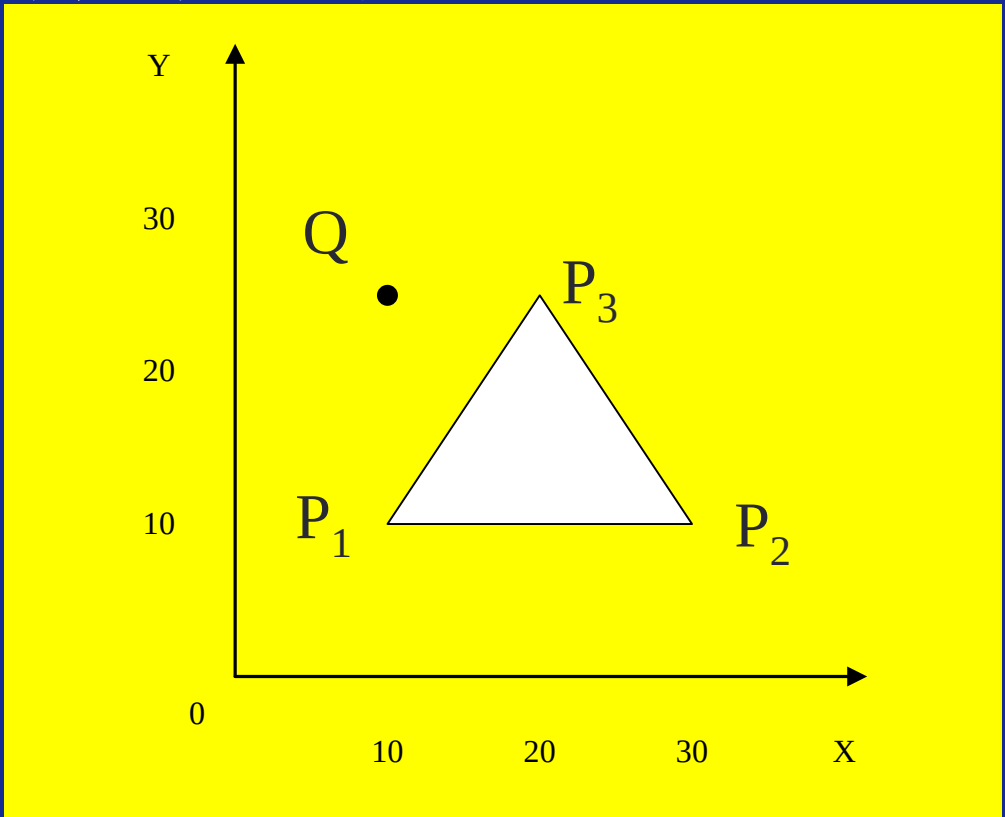
$$T_2 = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & -\cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

- (3)将A点移回到原位

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ x_A & y_A & 1 \end{bmatrix}$$

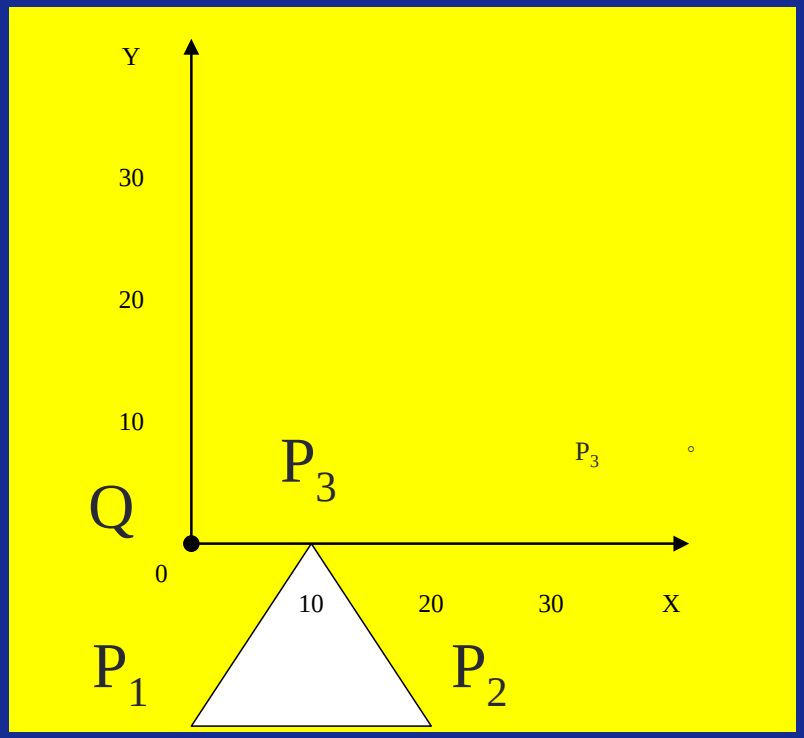
例题

- 例1 一个由顶点P1（10，10），P2（30，10）和P3（20，25）所定义的三角形，如图所示，相对于点Q（10，25）逆时针旋转30o，求变换后的三角形顶点坐标。



例题

- 第一步 Q点平移至坐标原点，如图所示。



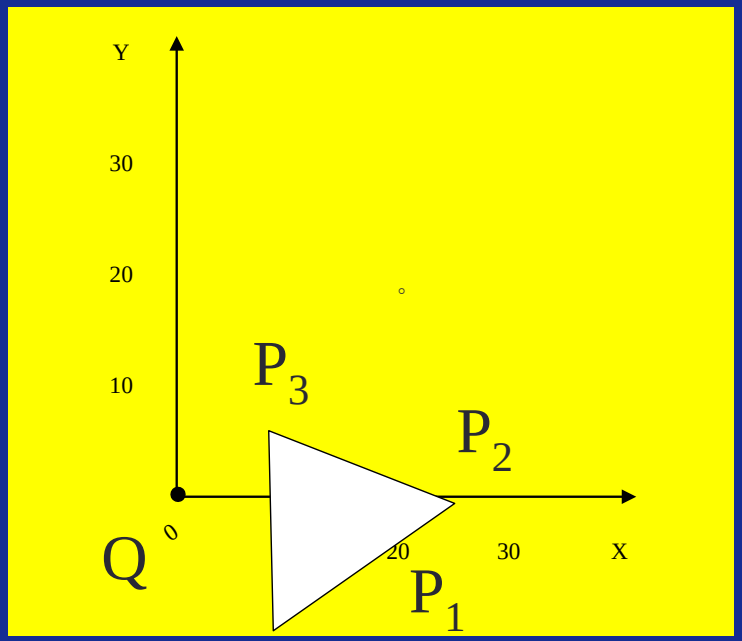
平移

变换矩阵为：

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -10 & -25 & 1 \end{bmatrix}$$

例题

- 第二步 三角形相对于坐标原点逆时针旋转30°，如图所示。



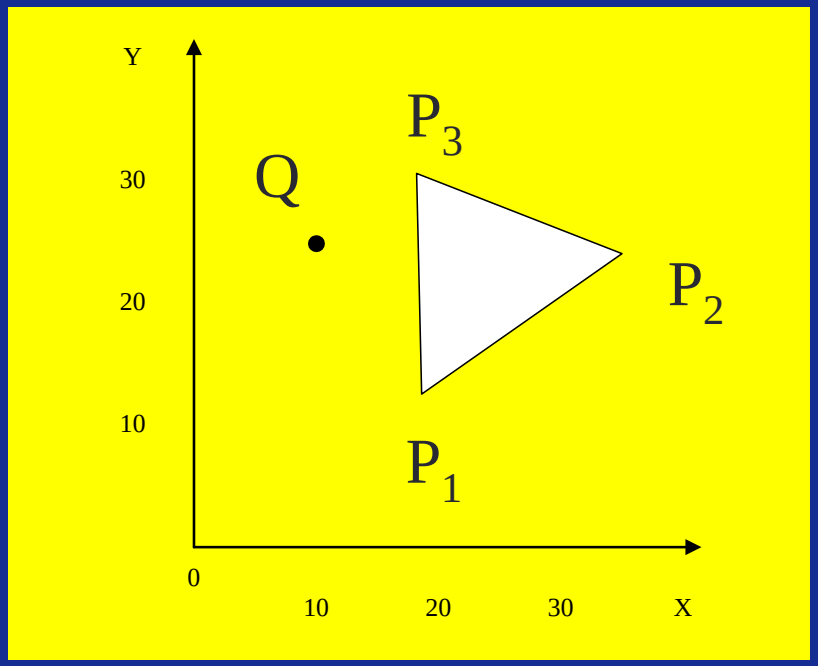
逆时针旋转

变换矩阵为：

$$T_2 = \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

例题

- 第三步 参考点Q平移回原位置，如图所示。



反平移

变换矩阵为：

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & 25 & 1 \end{bmatrix}$$

例题

- 图形变换后的顶点的规范化齐次坐标矩阵等于变换前的规范化齐次坐标矩阵乘以变换矩阵。

$$\begin{bmatrix} x_1' & y_1' & 1 \\ x_2' & y_2' & 1 \\ x_3' & y_3' & 1 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & 1 \\ x_2 & y_2 & 1 \\ x_3 & y_3 & 1 \end{bmatrix} \cdot T$$

而 $T = T_1 \cdot T_2 \cdot T_3$

所以

$$\begin{bmatrix} x_1' & y_1' & 1 \\ x_2' & y_2' & 1 \\ x_3' & y_3' & 1 \end{bmatrix} = \begin{bmatrix} 10 & 10 & 1 \\ 30 & 10 & 1 \\ 20 & 25 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -10 & -25 & 1 \end{bmatrix} \cdot \begin{bmatrix} \frac{\sqrt{3}}{2} & \frac{1}{2} & 0 \\ -\frac{1}{2} & \frac{\sqrt{3}}{2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 10 & 25 & 1 \end{bmatrix} = \begin{bmatrix} 17.5 & 12.01 & 1 \\ 34.82 & 22.01 & 1 \\ 18.66 & 30 & 1 \end{bmatrix}$$

例题

- 变换前的坐标点：

P1（10，10），P2（30，10）、P3（20，25）

- 变换后的坐标点：

P1（17.5，12.01），P2（34.82，22.01）、P3（18.66，30）

5.3.3 相对于任意方向的二维几何变换

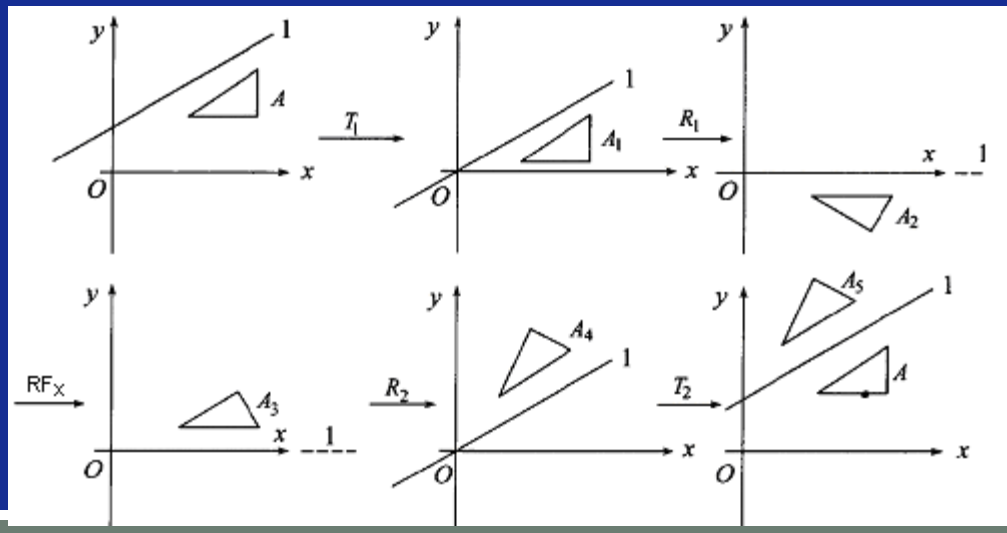
- 二维基本几何变换是相对于坐标轴进行的平移、比例、旋转、反射和错切这5种变换，但在实际应用中常会遇到变换方向不与坐标轴重合的情况。 相对于任意方向的变换方法：

  - 首先对任意方向做旋转变换，使该方向与坐标轴重合；
  - 然后对坐标轴进行二维基本几何变换；
  - 最后做反向旋转变换，将任意方向还原到原来的方向。

5.3.3 相对于任意方向的二维几何变换

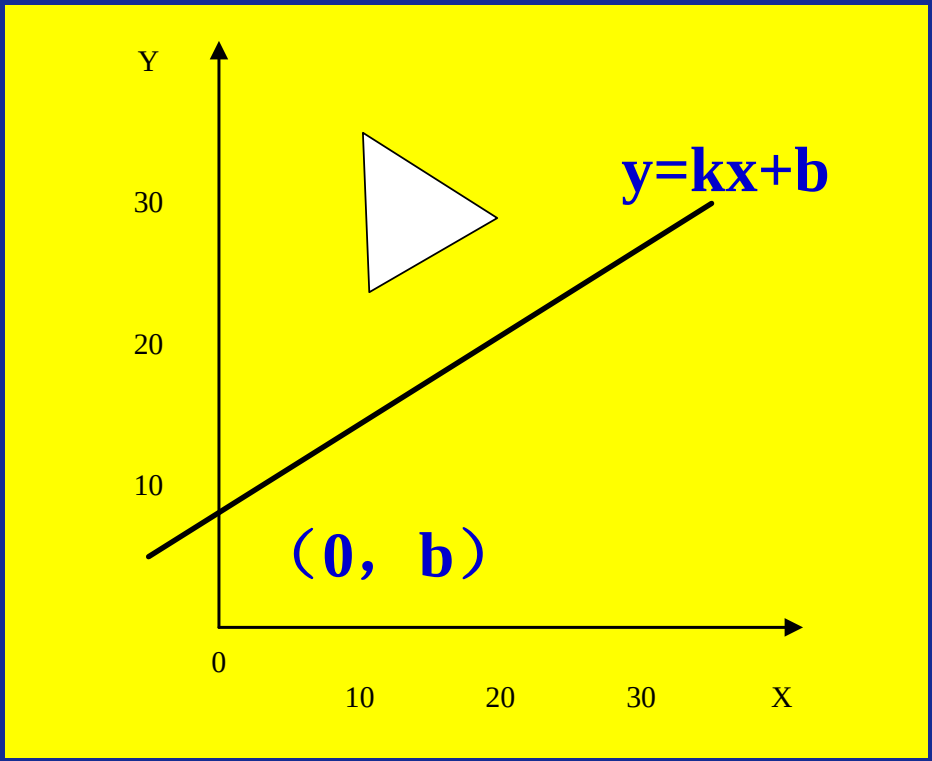
- 以任一直线L为对称轴的对称变换可以用变换合成的方法按如下步骤建立。
  - ①平移使直线L过坐标原点，记变换为T_1，图形A被变换到A_1。
  - ②旋转θ角，使直线L和ox轴重合，记变换为R_1，图形A_l被变换到A_2。
  - ③求图形A关于x轴的反射图形A_3，记变换为RF_x。
  - ④旋转-θ角，记变换为R_2，图形A_3被变换到A_4。
  - ⑤平移使直线L回到其原先的位置，记变换为T_2，图形A_4被变换到A_s。A_s即为A关于l的对称图形
- 总的变换为：

$$T_1 \cdot R_1 \cdot RF_x \cdot R_2 \cdot T_2$$



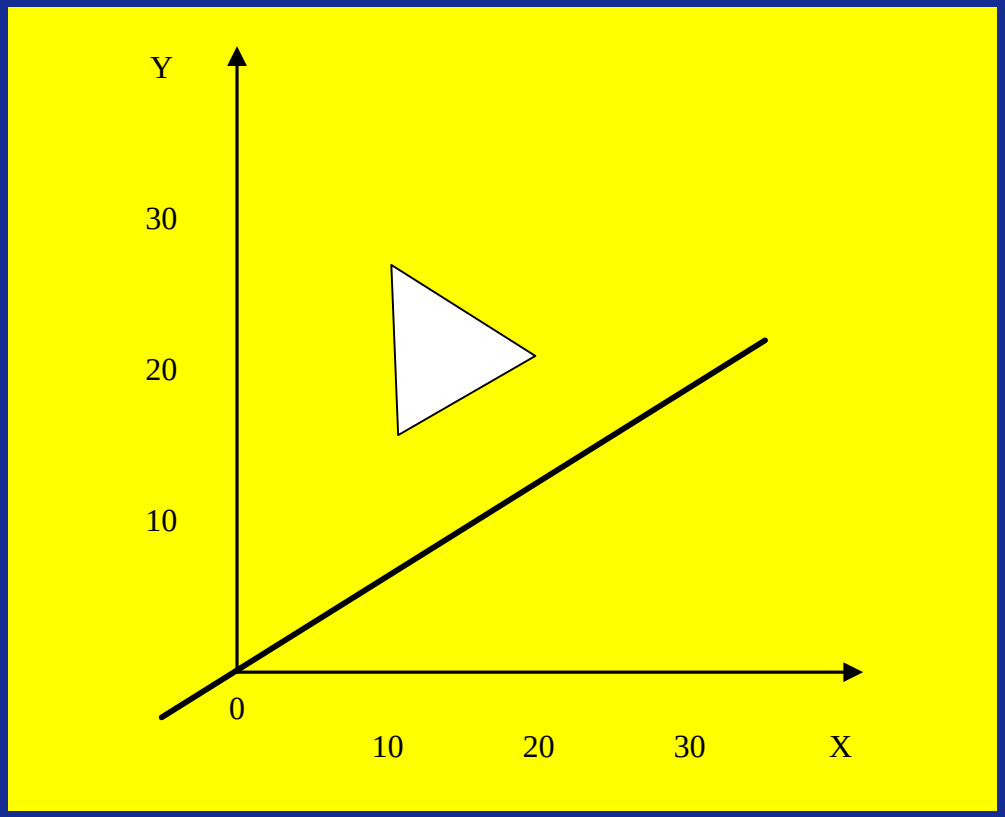
例题

- 例2 P1三角形相对于轴线y=kx+b作反射变换，求每一步相应的变换矩阵



例题

- 第一步 将点（0，b）平移至坐标原点，如图所示。



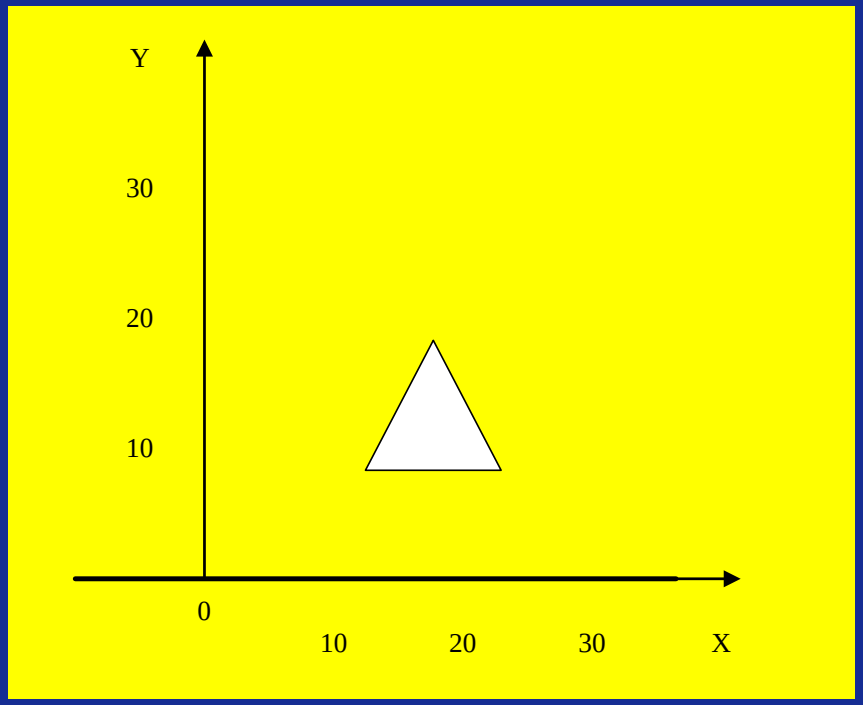
平移

变换矩阵为：

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & -b & 1 \end{bmatrix}$$

例题

- 第二步 将轴线y=kx绕坐标原点顺时针旋转β角(β=arctank)，落于x轴上，如图所示。



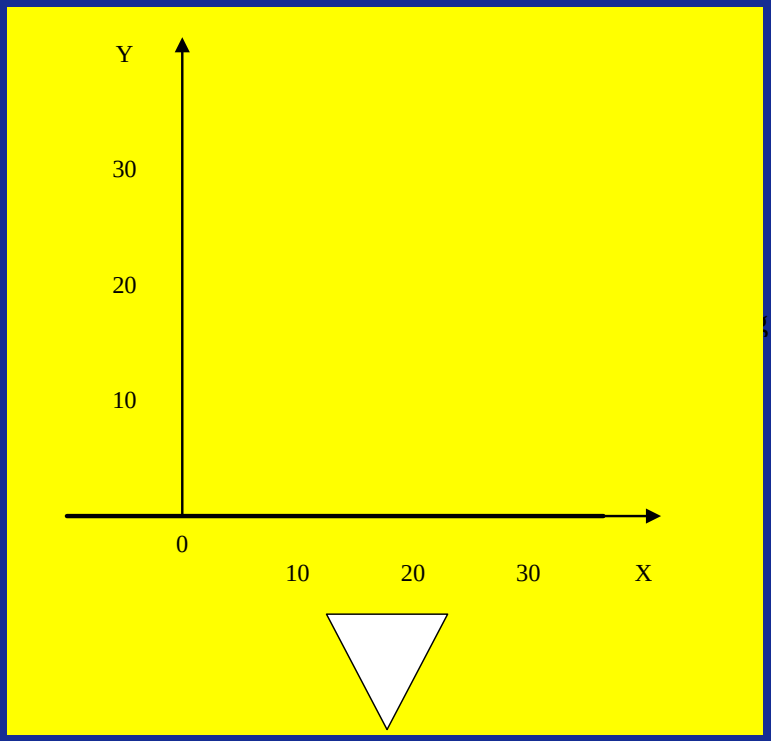
旋转

变换矩阵为：

$$T_2 = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

例题

- 第三步 三角形相对x轴作反射变换，如图所示。



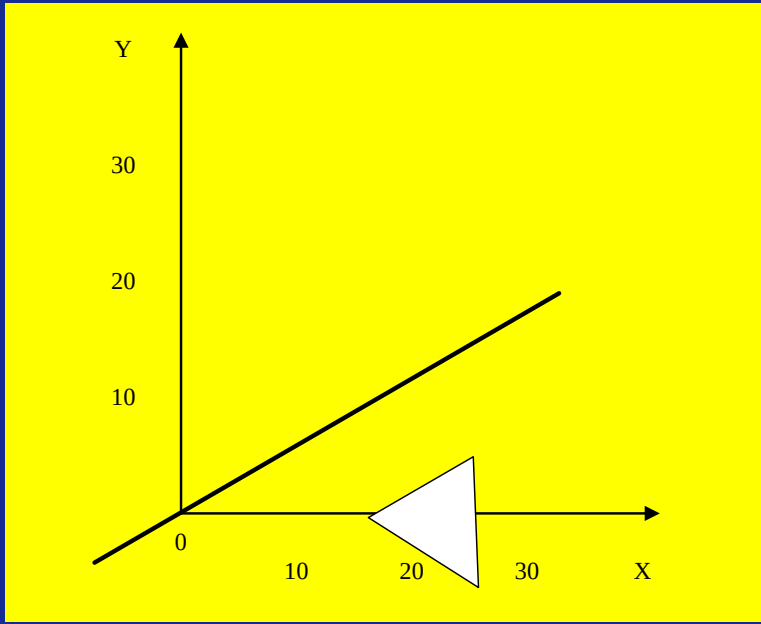
反射

变换矩阵为：

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

例题

- 第四步 将轴线y=kx逆时针旋转β角(β=arctank) ，如图所示。



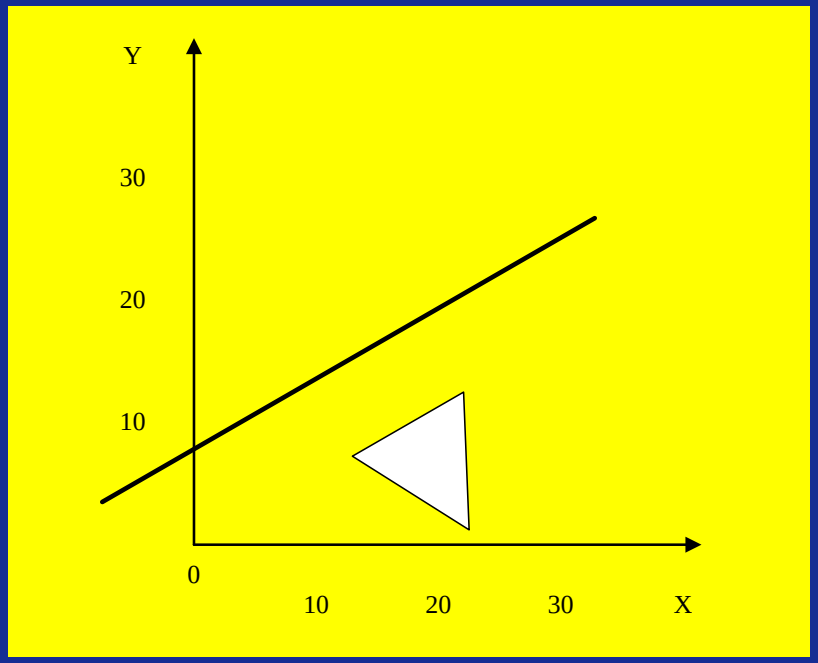
反旋转

变换矩阵为：

$$T_4 = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

例题

- 第五步 将轴线平移回原来的位置，如图所示。



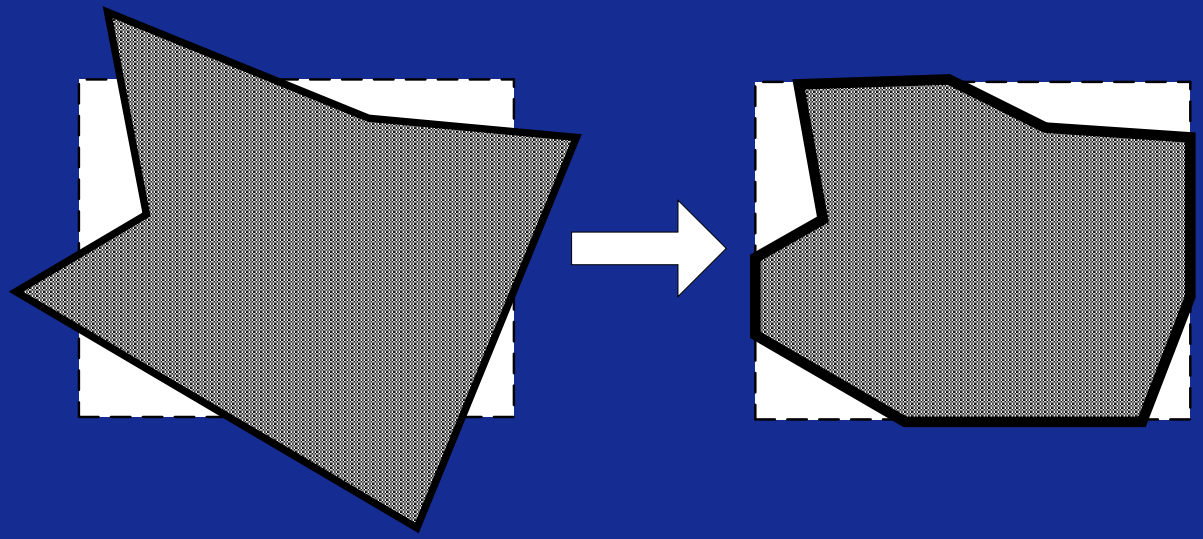
反平移

变换矩阵为：

$$T_5 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & b & 1 \end{bmatrix}$$

5.4 二维图形裁剪

- 裁剪：确定图形中哪些部分落在显示区之内，哪些落在显示区之外，以便只显示落在显示区内的那部分图形，这个选择过程称为裁剪。



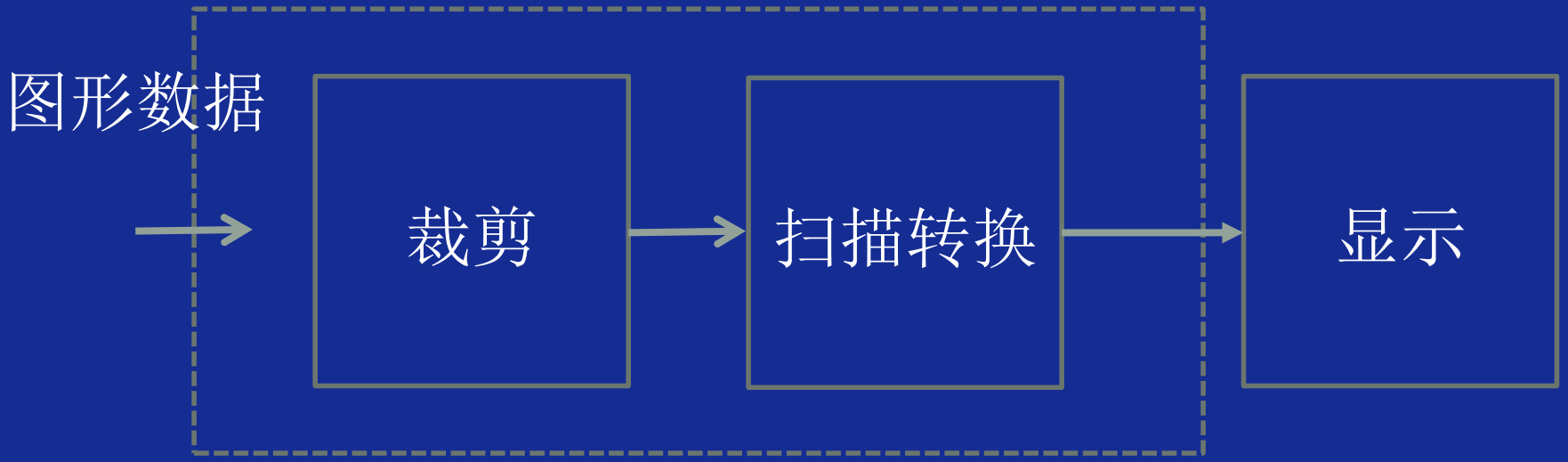
剪裁前　　剪裁后

与剪裁对应的显示区称为窗口

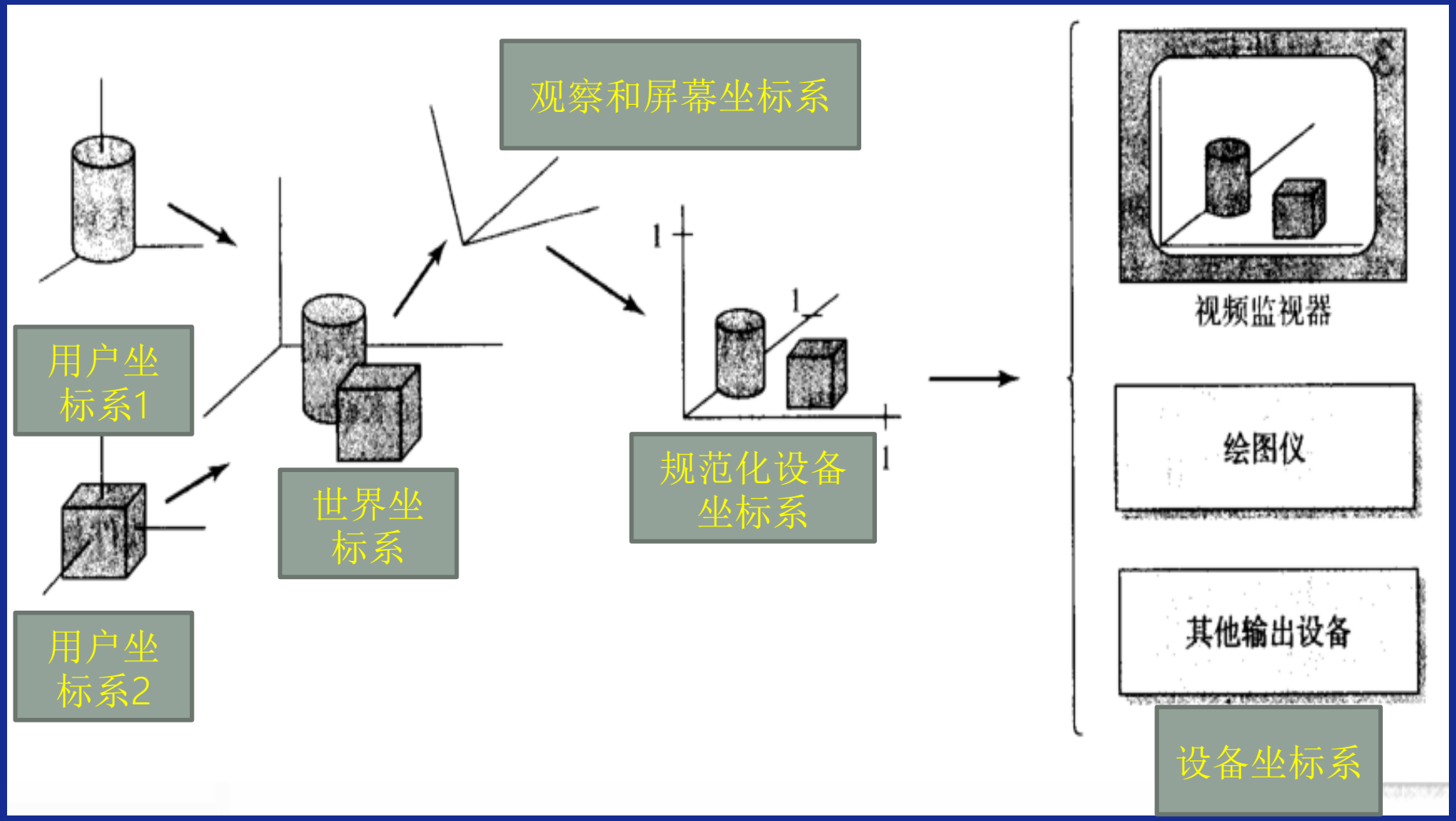
5.4 二维图形裁剪

图形显示前需要：扫描转换+裁剪

- 裁剪---〉扫描转换：最常用，节约计算时间。
- 扫描转换---〉裁剪：算法简单；

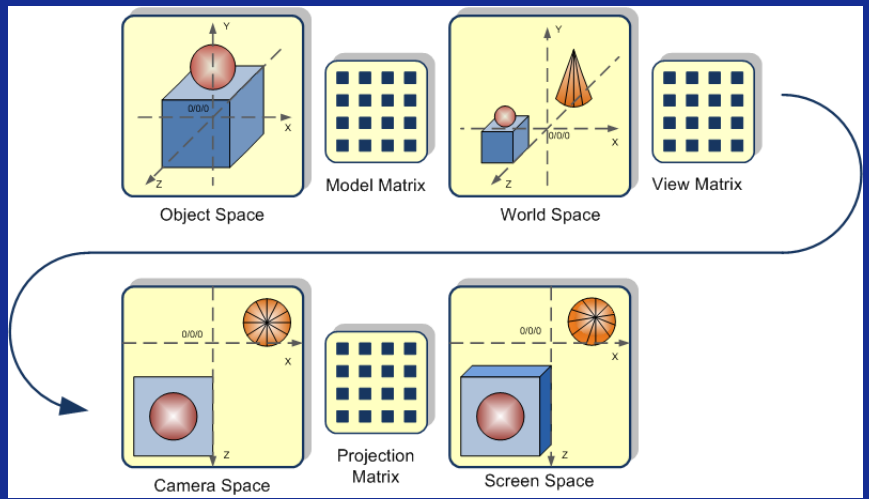


5.4.1 图形中常用的坐标系



5.4.1 图形中常用的坐标系

- 世界坐标系、用户坐标系、观察坐标系、屏幕坐标系和设备坐标系及规格化设备坐标系
- 用户坐标对象开始的坐标，相对于对象原点；将用户坐标转换为世界坐标，这些坐标是相对于世界的原点的；再将世界坐标转换为观察坐标，以摄像机或观察者的角度观察的坐标。在将坐标处理到观察空间之后，我们需要将其投影到屏幕坐标；最后将屏幕坐标转换为设备坐标。最后转换的坐标将会送到光栅器，由光栅器将其转化为片段。



5.4.1 图形中常用的坐标系

- 用户坐标系（局部坐标系，Local Coordinate System）

用于建立物体的几何模型，对象所在坐标空间。

在建立立方体几何模型时，用户坐标系原点放置在立方体的体心或立方体的一个角点上。

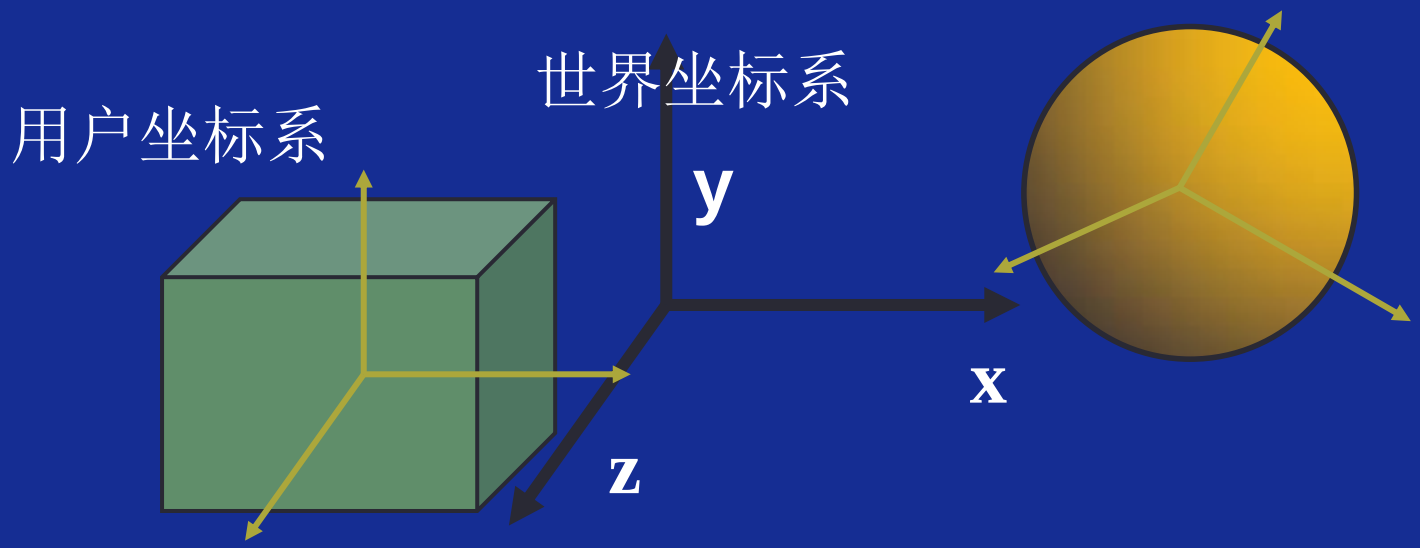
可能创建的所有模型都以（0，0，0）为起始位置，但他们会在世界空间的不同位置。

5.4.1 图形中常用的坐标系

■世界坐标系

描述现实世界中场景的固定坐标系。

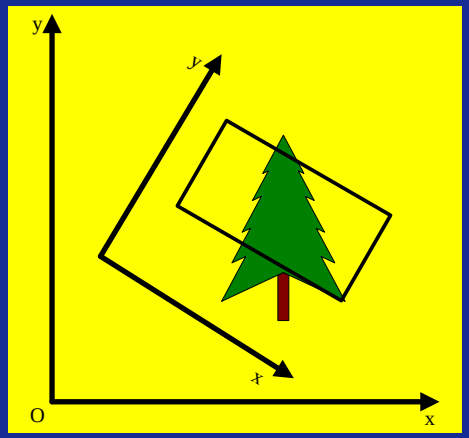
- 二维直角坐标系分为直角坐标系、极坐标系和球坐标系
- 三维直角坐标系可分为右手坐标系(OpenGL)和左手坐标系，确定z轴的方向。

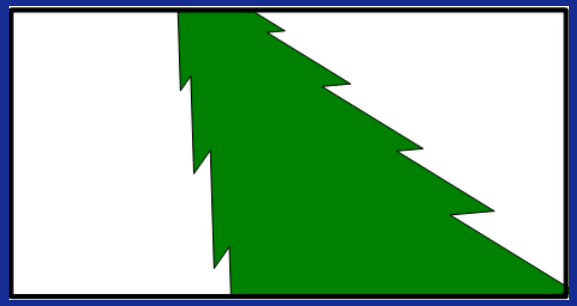


5.4.1 图形中常用的坐标系

观察坐标系

- 依据观察窗口的方向和形状在世界坐标系中定义的坐标系，主要用于指定图形的输出范围。
- 相应的观察空间被称为OpenGL的摄像机,就是从摄像机角度观察到的空间





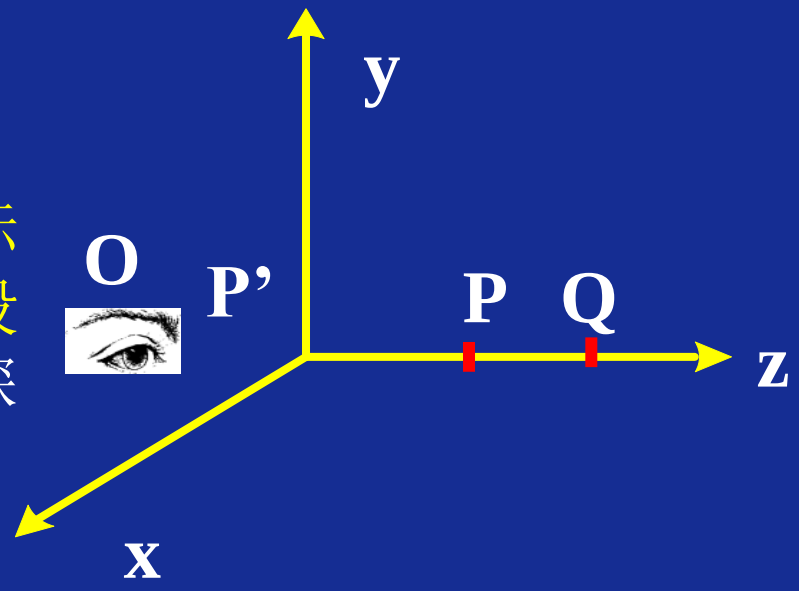
二维观察坐标系

5.4.1 图形中常用的坐标系

- 屏幕坐标系（Sceen Coordinates System）

- 又称投影坐标系，二维屏幕坐标系原点位于屏幕中心，x轴水平向右为正，y轴垂直向下为正。
- 三维屏幕坐标系是左手系，原点位于屏幕中心，z轴方向沿视线方向，x轴水平向右为正，y轴垂直向上为正。
-

只有三维屏幕坐标系才能正确反映投影时物体上点的深度信息

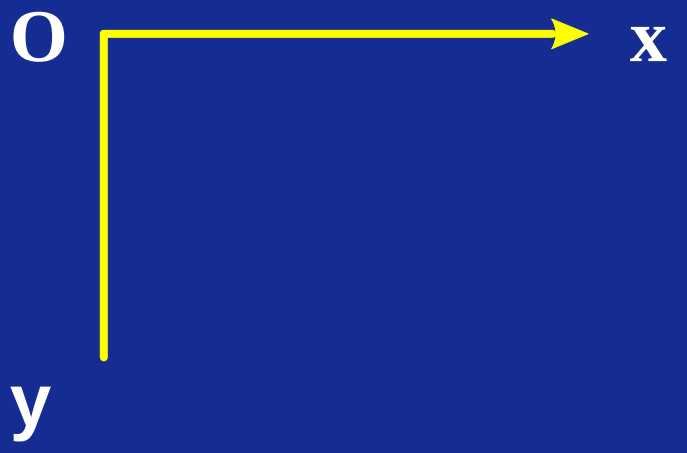


从视点O沿z方向的视线观察，P和Q点在屏幕上的投影都在P′，但P、Q与视点的距离不同，P点离视点近，应遮挡住Q点。

5.4.1 图形中常用的坐标系

- 设备坐标系（Device Coordinates）

与图形显示设备相关的坐标系。如显示器或打印机的坐标系。原点位于屏幕客户区的左上角，x轴水平方向为正，y轴垂直向下为正，基本单位为像素。



设备坐标系

规格化设备坐标系

5.4.1 图形中常用的坐标系

- 规范化设备坐标系（Normal Device Coordinates ）
  - 规格化设备坐标系是将设备坐标系规格化到（0.0，0.0）到（1.0，1.0）的范围内而定义的坐标系。
  - 规格化设备坐标系独立于具体输出设备。一旦图形变换到规格化设备坐标系中，只要作一个简单的乘法运算即可映射到具体的设备坐标系中。
  - 由于规格化设备坐标系能统一用户各种图形的显示范围，故把用户图形变换成规格化设备坐标系中的统一大小标准图形的过程叫作图形的逻辑输出。把规格化设备坐标系中的标准图形送到显示设备上输出的过程叫作图形的物理输出。有了规格化设备坐标系后，图形的输出可以在抽象的显示设备上进行讨论，因而这种图形学又称为与具体设备无关的图形学。

5.4.2 窗口与视区

- 在观察坐标系中定义的确定显示内容的区域称为窗口。显然此时窗口内的图形是用户希望在屏幕上输出的，窗口是裁剪图形的标准参照物。
- 在屏幕坐标系中定义的输出图形的区域称为视区。
- 视区和窗口的大小可以不相同。一般情况下，用户把窗口内感兴趣的图形输出到屏幕上相应的视区内。在屏幕上可以定义多个视区，用来同时显示不同的窗口内的图形信息。

5.4.2 窗口与视区

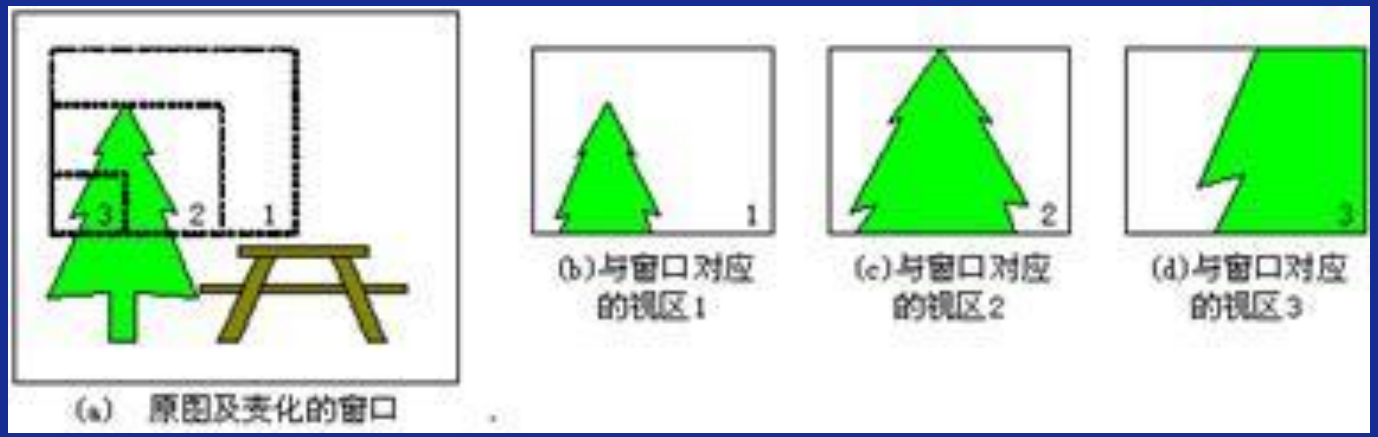
定义三个窗口

显示三个视区

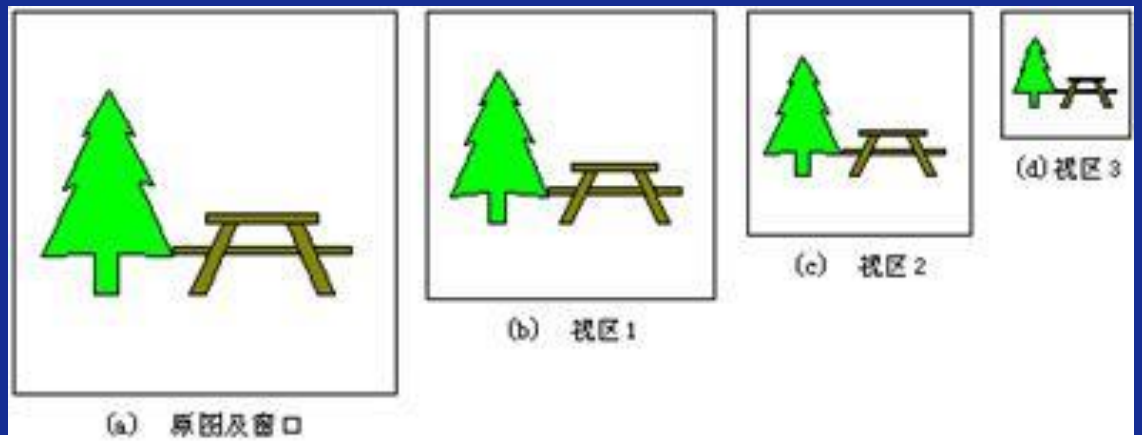
- 图形输出需要进行从窗口到视区的变换，只有窗口内的图形才能在视区中输出，并且输出的形状要根据视区的大小进行调整，这称为窗视变换（Window Viewport Transformation，WVT）。在二维图形观察中，可以这样理解，窗口相当于一个一扇窗户，窗口内的图形是希望看到的，就在视区中输出，窗口外的图形不希望看到，不在视区中输出，因此需要对窗口中输出的二维图形进行裁剪。

5.4.2 窗口与视区

- 变焦距效果:将不同尺寸的裁剪窗口连续映射到固定尺寸的视口

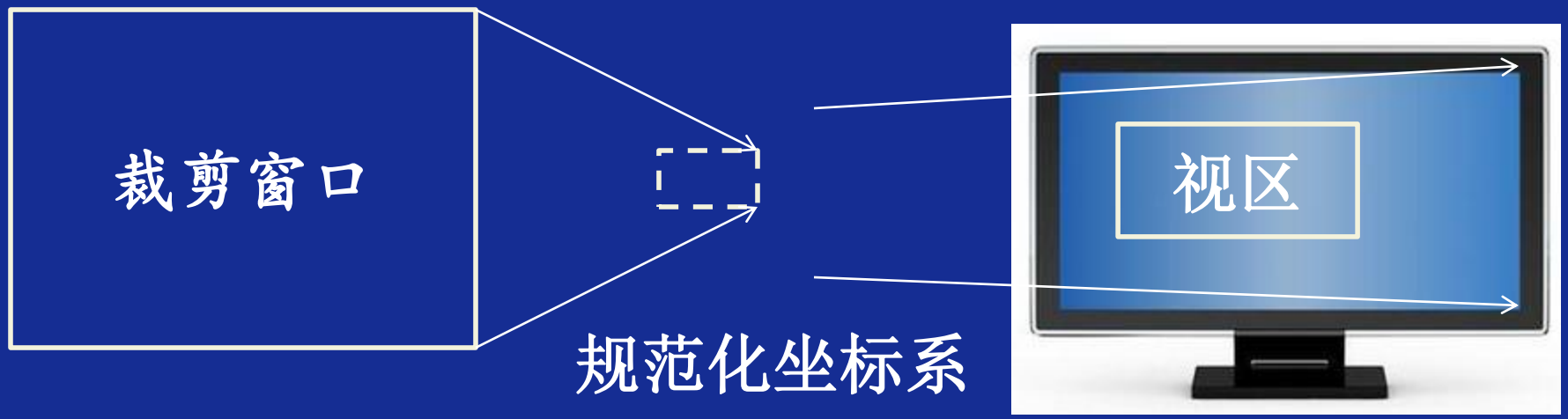


- 缩放效果:固定尺寸的裁剪窗口映射到不同尺寸的视口



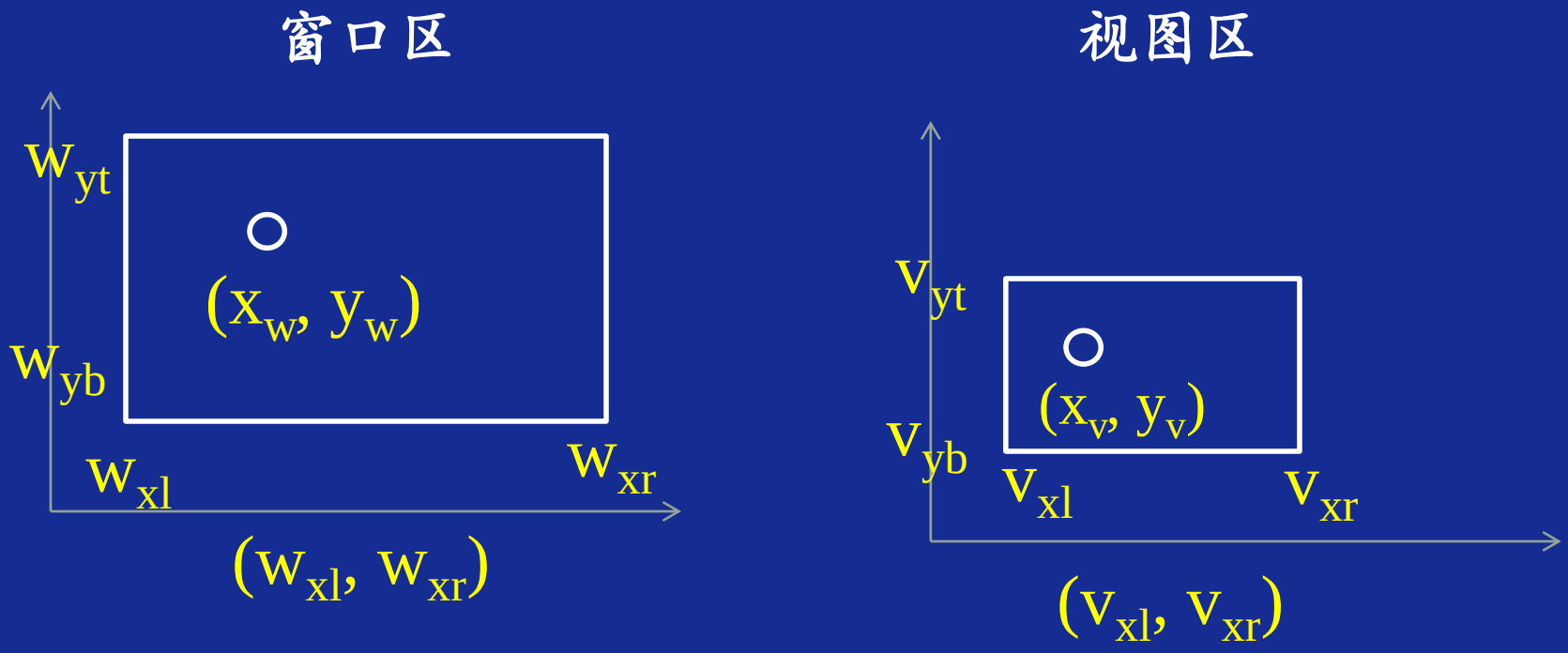
窗视变换

- 图形输出需要进行从窗口到视区的变换,只有窗口内的图形才能在视区中输出,并且输出的行只要根据视区的大小进行调整,这种变换称为窗视变换(Window Viewport Transformation ,WVT).



用户只希望看到窗口内的的图形，并在视区中输出，窗口外的不希望被看到，因此需要对窗口中输出的二维图形进行裁剪

窗口区到视图区的坐标变换



假定把窗口内的一点 (x_w, y_w) 转换为视区中的一点 (x_v, y_v) 这属于相对于任意一个参考点的二维几何变换，变换步骤如下

坐标系

- （1）将窗口的左下角 (w_{xl}, w_{yb}) 移到观察坐标系原点，平移参数为 $(-w_{xl}, -w_{yb})$

$$T_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -w_{xl} & -w_{yb} & 1 \end{bmatrix}$$

坐标系

- （2）对原点进行比例变换，使窗口的大小与视区大小相等，即将窗口变为视区

$$T_2 = \begin{bmatrix} S_x & 0 & 0 \\ 0 & S_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$s_x = \frac{v_{xt} - v_{xl}}{w_{xt} - w_{xl}}$$

$$s_y = \frac{v_{yt} - v_{yl}}{w_{yt} - w_{yl}}$$

坐标系

- （3）进行反平移，将视区的左下角点平移到设备坐标系的 (v_{xl}, v_{yb})

$$T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_{xl} & v_{yb} & 1 \end{bmatrix}$$

坐标系

- 因此，窗视变换矩阵

$$T = T_1 g T_2 g T_3 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ -w_{xl} & -w_{yb} & 1 \end{bmatrix} \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ v_{xl} & v_{yb} & 1 \end{bmatrix}$$

$$[x_v \quad y_v \quad 1] = [x_w \quad y_w \quad 1] \cdot \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ v_{xl} - w_{xl} s_x & v_{yb} - w_{yb} s_y & 1 \end{bmatrix}$$

坐标系

- 写成方程

$$\begin{cases} x_v = S_x x_w + v_{xl} - w_{xl} S_x \\ y_v = S_y y_w + v_{yb} - w_{yb} S_y \end{cases}$$

$$\begin{cases} a = S_x \\ b = v_{xl} - w_{xl} S_x \\ c = S_y \\ d = v_{yb} - w_{yb} S_y \end{cases}$$

则窗视变换的展开式为

$$\begin{cases} x_v = a x_w + b \\ y_v = c y_w + d \end{cases}$$

5.5 剪裁

- 在二维观察中，需要在观察坐标系下根据窗口大小对世界坐标系中的二维图形进行裁剪（clipping），只将位于窗口内的图形变换到视区输出。直线的裁剪是二维图形裁剪的基础，裁剪的实质是判断直线是否与窗口相交，如相交则进一步确定位于窗口内的部分

5.5 直线段剪裁

直线段

直线段裁剪算法是二维图形裁剪的基础。复杂的曲线可以通过折线段来近似，从而裁剪问题也可以化为直线段的裁剪问题。直线段的裁剪实质是判断直线段是否与窗口边界有交点，如相交则进一步确定直线段位于窗口内的部分。

常用线段裁剪方法包括:

(1)Cohen-Sutherland法

(2)中点分割法

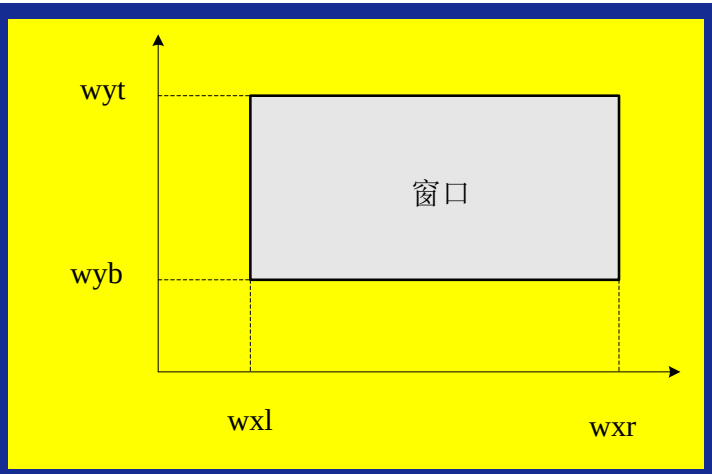
(3)梁友栋-Barskey法

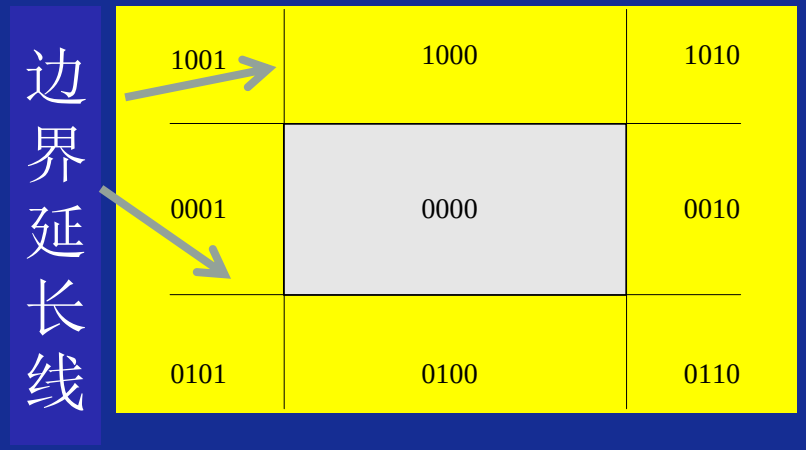
Cohen-Sutherland裁剪算法

- 1、给出直线段的端点，确定端点编码；
- 2、判断直线段是否在裁剪区域内，是否位于窗外的同一侧；
- 3、直线段与左、右、上、下边界的交点；
- 4、根据"简弃"原则，去掉边界外的直线段。

Cohen-Sutherland裁剪算法

- Cohen-Sutherland直线裁剪算法是最早流行的编码算法。每条直线的端点都被赋予一组四位二进制代码，称为区域编码(Region Code，RC)，用来标识直线端点相对于窗口边界及其延长线的位置。
- 假设窗口是标准矩形，由上（y=wyt）、下（y=wyb）、左（x=wxl）、右（x=wxr）四条边组成。延长窗口四条边形成9个区域。这样根据直线的任一端点P(x，y)所处的窗口区域位置，可以赋予一组4位二进制区域码$RC=C_3C_2C_1C_0$。





- 为了保证窗口内直线端点的编码为零，编码规则定义如下：
- 第一位：若端点位于窗口之左侧，即$x<wxl$，则$C_0=1$，否则$C_0=0$。
- 第二位：若端点位于窗口之右侧，即$x>wxr$，则$C_1=1$，否则$C_1=0$。
- 第三位：若端点位于窗口之下侧，即$y<wyb$，则$C_2=1$，否则$C_2=0$。
- 第四位：若端点位于窗口之上侧，即$y>wyt$，则$C_3=1$，否则$C_3=0$。

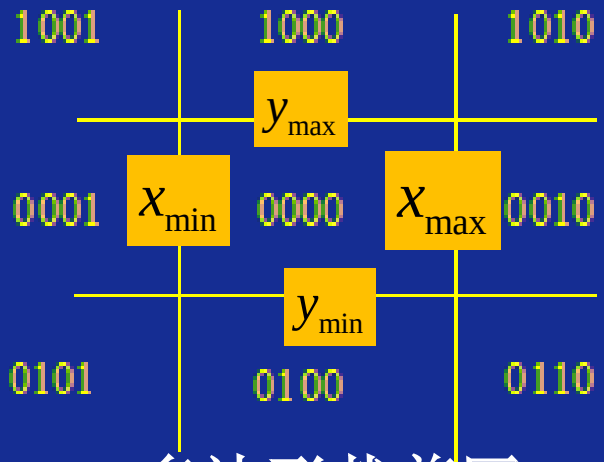
Cohen-Sutherland裁剪算法

- 判断一条直线段与窗口属何种关系

采用如下编码：延长窗口的边，将二维平面分成9个区域，每个区域赋予4位编码 $C_tC_bC_rC_l$ ，其中各位编码的定义如下

$$C_t=\begin{cases}1 & y>y_{max}\\ 0 & other\end{cases}\qquad C_b=\begin{cases}1 & y<y_{min}\\ 0 & other\end{cases}$$

$$C_r=\begin{cases}1 & x>x_{max}\\ 0 & other\end{cases}\qquad C_l=\begin{cases}1 & x>x_{min}\end{cases}$$



多边形裁剪区域编码

5.5.2 Cohen-Sutherland裁剪步骤

- 若直线的两个端点的区域编码都为零，即$RC_1|RC_2=0$（二者按位相或的结果为零，即$RC_1=0$且$RC_2=0$），说明直线两端点都在窗口内，应“简取”之。
- 若直线的两个端点的区域编码都不为零，即$RC_1\&RC_2\neq0$（二者按位相与的结果不为零，即$RC_1\neq0$且$RC_2\neq0$，即直线位于窗外的同一侧，说明直线的两个端点都在窗口外，应“简弃”之。

5.5.2 Cohen-Sutherland裁剪步骤

- 若直线既不满足“简取”也不满足“简弃”的条件，直线必然与窗口相交，需要计算直线与窗口边界的交点。交点将直线分为两段，其中一段完全位于窗口外，可“简弃”之。对另一段赋予交点处的区域编码，再次测试，再次求交，直至找到确定完全位于窗口内的直线段为止。

　　实现时，一般按固定顺序左（x=wxl），右（x=wxr）、下（y=wyb）、上（y=wyt）求解窗口与直线的交点。

5.5.3 交点计算公式

- 求交

假定直线的端点坐标为(x0, y0)和(x1, y1)

(1)左、右边界交点的计算：y=y0+k(x - x0)；

其中：x是窗口左边界或右边界的值

(2)上、下边界交点的计算：x=x0+(y - y0)/k。

其中：y是窗口下边界或上边界的值

5.6 中点分割裁剪算法

基本思想

与前一种Cohen-Sutherland算法一样首先对线段端点进行编码，并把线段与窗口的关系分为三种情况:

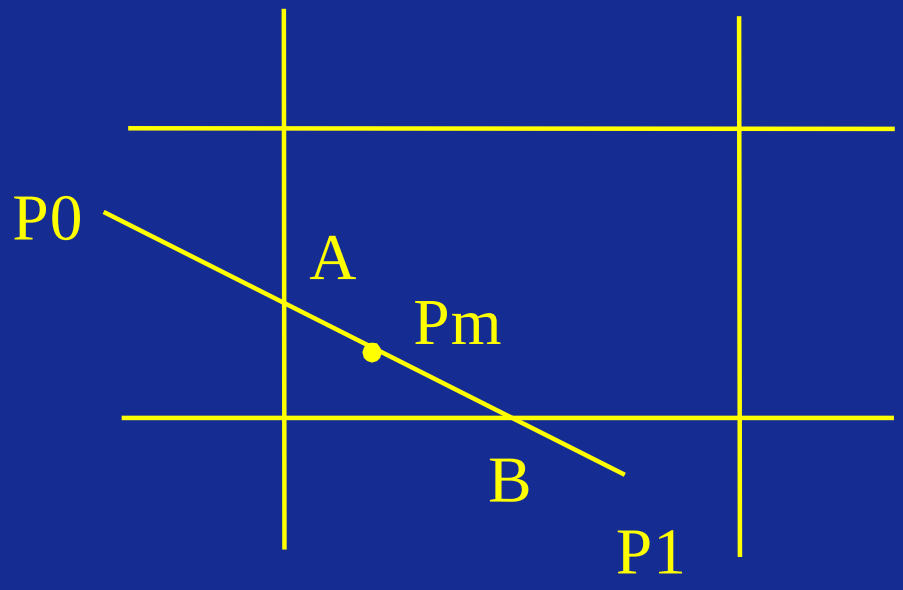
全在窗口内、完全不在窗口内和线段与窗口有交。

对前两种情况，进行一样的处理。

对于第三种情况，不需要求解直线段与窗口边界的交点就可以对直线段进行裁剪。

中点分割裁剪算法

- **基本思想**：采用二分法的思想来逐次计算直线的的中点以逼近窗口边界



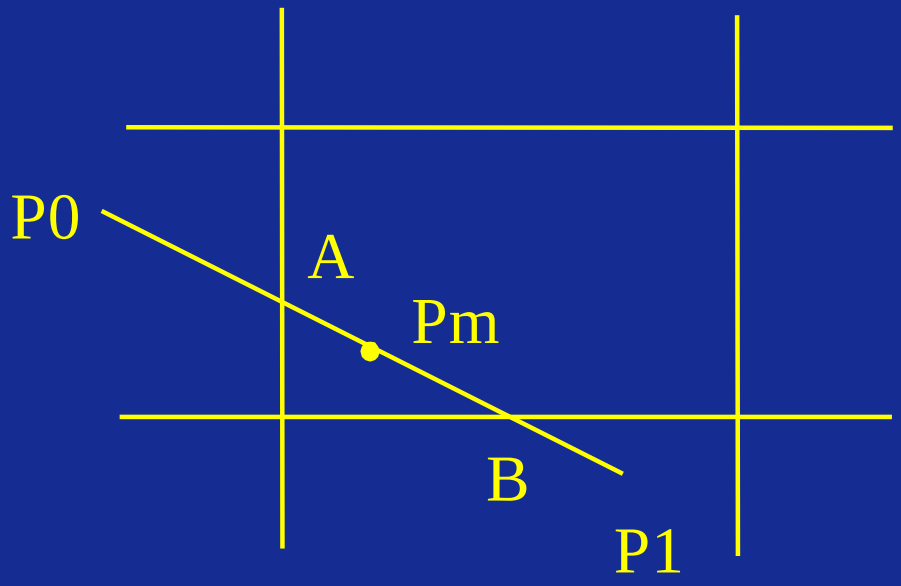
线段中点分割裁剪

从 P_0 出发找最近可见点的方法

先求出 P_0P_1 的中点 P_m

若 P_0P_m 不是显然不可见的，并且 P_0P_m 在窗口中有可见部分，则距 P_0 最近的可见点一定落在 P_0P_m 上，所以用 P_0P_m 代替 P_0P_1；

中点分割裁剪算法



线段中点分割裁剪

否则取 P_mP_1 代替 P_0P_1 再对新的 P_0P_1 求中点 P_m 。重复上述过程，直到 P_mP_1 长度小于给定的控制常数为止，此时 P_m 收敛于交点。

中点分割裁剪算法

(1)若code1|code2=0，对直线段应简取之，结束；否则，若code1&code2≠0，对直线段可简弃之，结束；当这两条均不满足时，进行步骤(2)。

(2)找出该直线段离窗口边界最远的点和该直线段的中点。判中点是否在窗口内：若中点不在窗口内，则把中点和离窗口边界最远点构成的线段丢掉，以线段上的另一点和该中点再构成线段求其中点；如中点在窗口内，则又以中点和最远点构成线段，并求其中点，直到中点与窗口边界的坐标值在规定的误差范围内相等，则该中点就是该线段落在窗口内的一个端点坐标。

(3)如另一点在窗口内，则经(2)即确定了该线段在窗口内的部分。如另一点不在窗口内，则该点和所求出的在窗口上的那一点构成一条线段，重复步骤(2)，即可求出落在窗口内的另一点。

第四章测评

- 多边形的表示方法有哪两种？分别简单解释这两种表示方法。
- 什么是多边形的扫描转换？简述扫描线填充算法步骤。
- 有效边表简述扫描线求交时顶点的处理原则
- 多边形填充过程中边界像素的处理原则是什么？

第四章测评答案

- 多边形的表示方法有哪两种？分别简单解释这两种表示方法。

答：多边形的表示包括顶点表示法和点阵表示法。两种表示的解释详见书P112.

- 什么是多边形的扫描转换？简述扫描线算法步骤。

答：将多边形的描述从顶点表示法变换到点阵表示法的过程，称为多边形的扫描转换。扫描线算法步骤见书P113.

第四章测评答案

- 有效边表简述扫描线求交时顶点的处理原则。

答：普通连接点的顶点个数计数为1；局部最低点的顶点个数计数为2；局部最高点的顶点个数计数为0.

- 多边形填充过程中边界像素的处理原则是什么？

答："左闭右开"和"下闭上开".

梁友栋-Barskey裁剪算法

- 梁友栋和Barsky提出了比Cohen - Sutherland算法速度更快的直线段裁剪算法。
- 该算法是以直线段的参数方程为基础设计的，把判断直线段与窗口边界求交的二维裁剪问题转化为求解一组不等式，确定直线段参数的一维裁剪问题。
- 梁友栋 - Barsky算法把直线段和窗口的相互位置关系划分为两种情况进行讨论：平行于窗口边界的直线段和不平行于窗口边界的直线段。

梁友栋-Barskey裁剪算法

- 设直线段的参数方程为：

$$\begin{cases} x = x_0 + t(x_1 - x_0) \\ y = y_0 + t(y_1 - y_0) \end{cases}$$

式中，0≤t≤1

- 对于对角点坐标为（W_{xl}，W_{yb}）、（W_{xr}，W_{yt}）的裁剪窗口，直线段裁剪条件为：
-

$$\begin{aligned} w_{xl} &\le x_0 + t(x_1 - x_0) \le w_{xr} \\ w_{yb} &\le y_0 + t(y_1 - y_1) \le w_{yt} \end{aligned}$$

梁友栋-Barskey裁剪算法

- 分解后

$$\begin{cases} t(x_0 - x_1) \le x_0 - w_{xl} \\ t(x_1 - x_0) \le w_{xr} - x_0 \\ t(y_0 - y_1) \le y_0 - w_{yb} \\ t(y_1 - y_0) \le w_{yt} - y_0 \end{cases}$$

- 令：

$$\Delta x = x_1 - x_0, \Delta y = y_1 - y_0$$

$$u_1 = -\Delta x, u_2 = \Delta x, u_3 = -\Delta y, u_4 = \Delta y$$

$$v_1 = x_0 - w_{xl}, v_1 = w_{xr} - x_0, v_3 = y_0 - w_{yb}, v_4 = w_{yt} - y_0$$

梁友栋-Barskey裁剪算法

- 则

$$t \cdot u_n \le v_n$$

其中，n=1，2，3，4。

上式给出了直线段的参数方程裁剪条件。

n代表直线段裁剪时，窗口的边界顺序，

n = 1表示左边界；n = 2表示右边界；

n = 3表示下边界；n = 4表示上边界。

梁友栋-Barskey裁剪算法

■梁友栋－Barsky算法主要考察直线段方程参数t的变化情况。为此，先讨论直线段和窗口边界不平行的情况。

$$t_n = \frac{v_n}{u_n}, u_n \neq 0$$

从上式可以知道，x1≠x2而且y1≠y2，这意味着直线段不和窗口的任何边界平行，直线段及其延长线和窗口边界及其延长线必定相交，可以采用参数t对直线进行裁剪。

梁友栋-Barskey裁剪算法

- 假定，直线段L1的起点坐标为(x_0, y_0)，终点坐标为(x_1, y_1)。

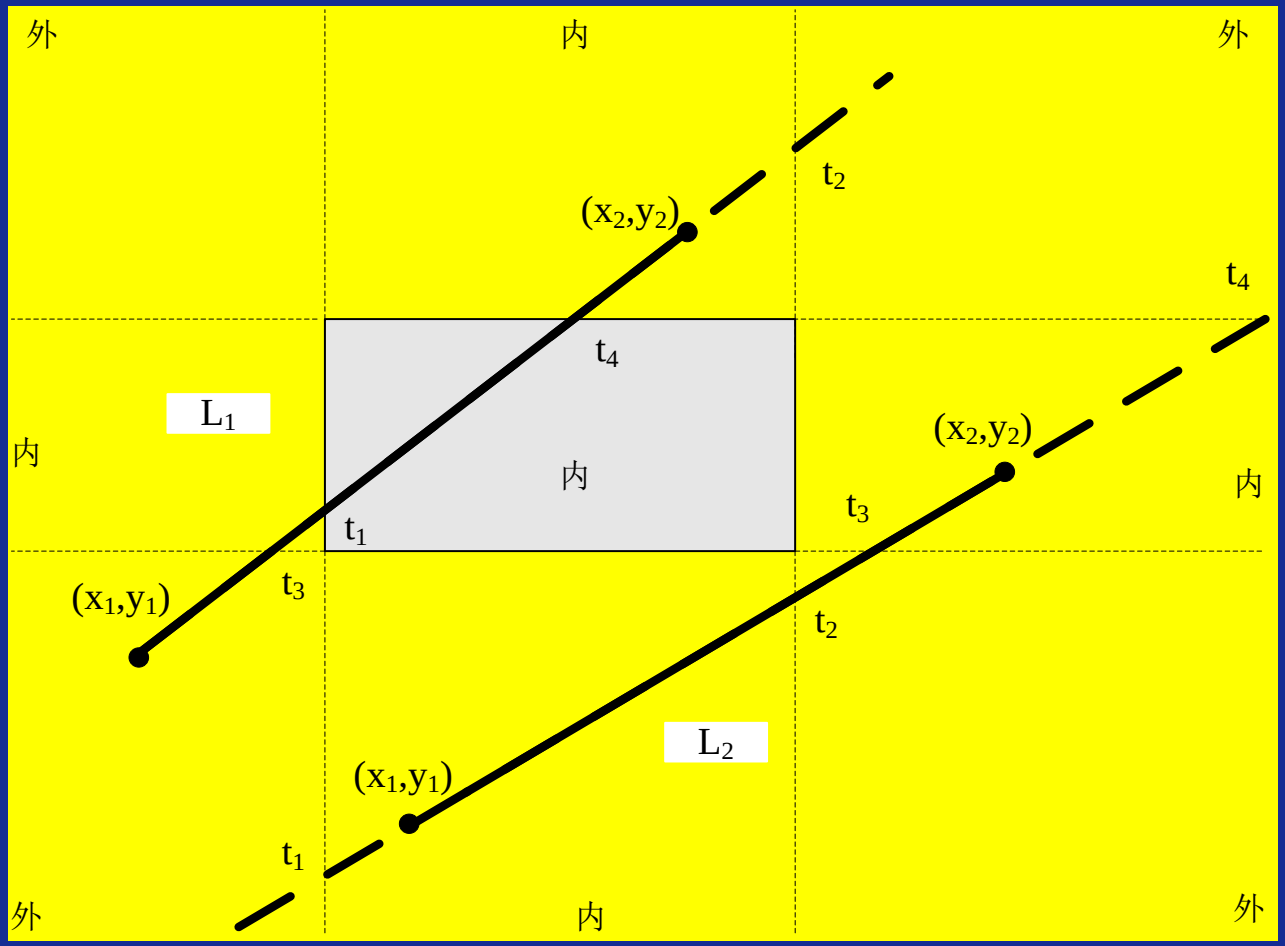
$$u_n < 0$$

- 表示在该处直线段从裁剪窗口及其边界延长线的外部延伸到窗口内部，直线段和窗口边界的交点位于直线段的起点一侧；

$$u_n > 0$$

- 表示在该处直线段从裁剪窗口及其边界延长线的内部延伸到窗口外部，直线段和窗口边界的交点位于直线段的终点一侧。

梁友栋-Barskey裁剪算法



梁友栋-Barskey裁剪算法

- $u_1 < 0$ 时，表示直线段从窗口左边界的外部延伸到窗口内部，和窗口左边界及其延长线相交于参数t等于t1处；
- $u_2 > 0$ 时，表示直线段从窗口右边界的内部延伸到窗口外部，和窗口右边界及其延长线相交于参数t等于t2处；
- $u_3 < 0$ 时，表示直线段从窗口下边界的外部延伸到窗口内部，和窗口下边界及其延长线相交于参数t等于t3处
- $u_4 > 0$ 时，表示直线段从窗口上边界的内部延伸到窗口外部，和窗口上边界及其延长线相交于参数t等于t4处。

L1与裁剪窗口的交点参数是t1和t4。

- 直线段的起点判断：对于直线段的起点一侧，t1>t3，当 $u_n < 0$时，被裁剪直线段的起点取t的最大值，$t_{max} = t_1$;
- 直线段的终点判断：对于直线段的起点一侧，t4<t2，当 $u_n > 0$时，被裁剪直线段的起点取t的最小值，$t_{\min} = t_4$;

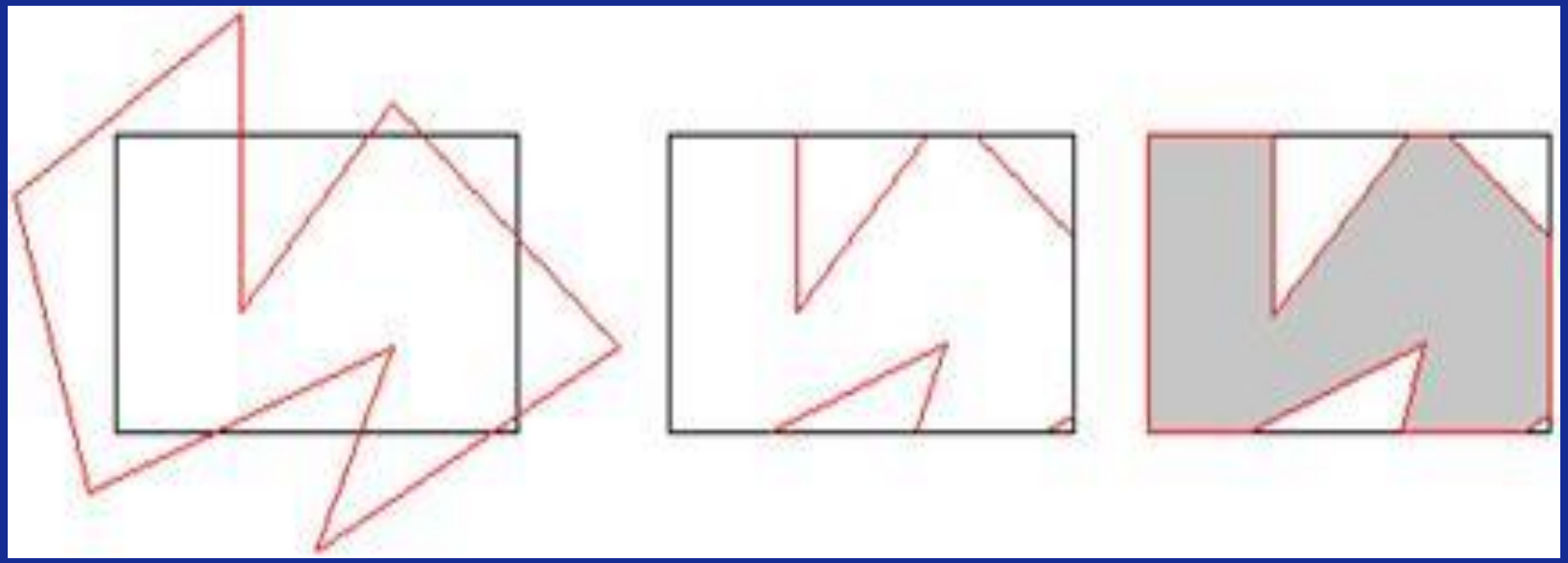
如果$t_{max} \le t_{min}$，则被裁剪窗口的直线段位于窗口内

思考

利用liang-Barsky直线段裁剪算法对直线段L2的裁剪过程

5.8 多边形裁剪(重点)

- 直接用直线段裁剪算法对多边形的每条边进行裁剪，会得到如下图b的结果，而实际结果应该如下图c所示。



（a）裁剪前　（b）直接采用直线段裁剪的结果　（c）正确的裁剪结果

5.8 多边形裁剪(重点)

Sutherland-Hodgman基本思想

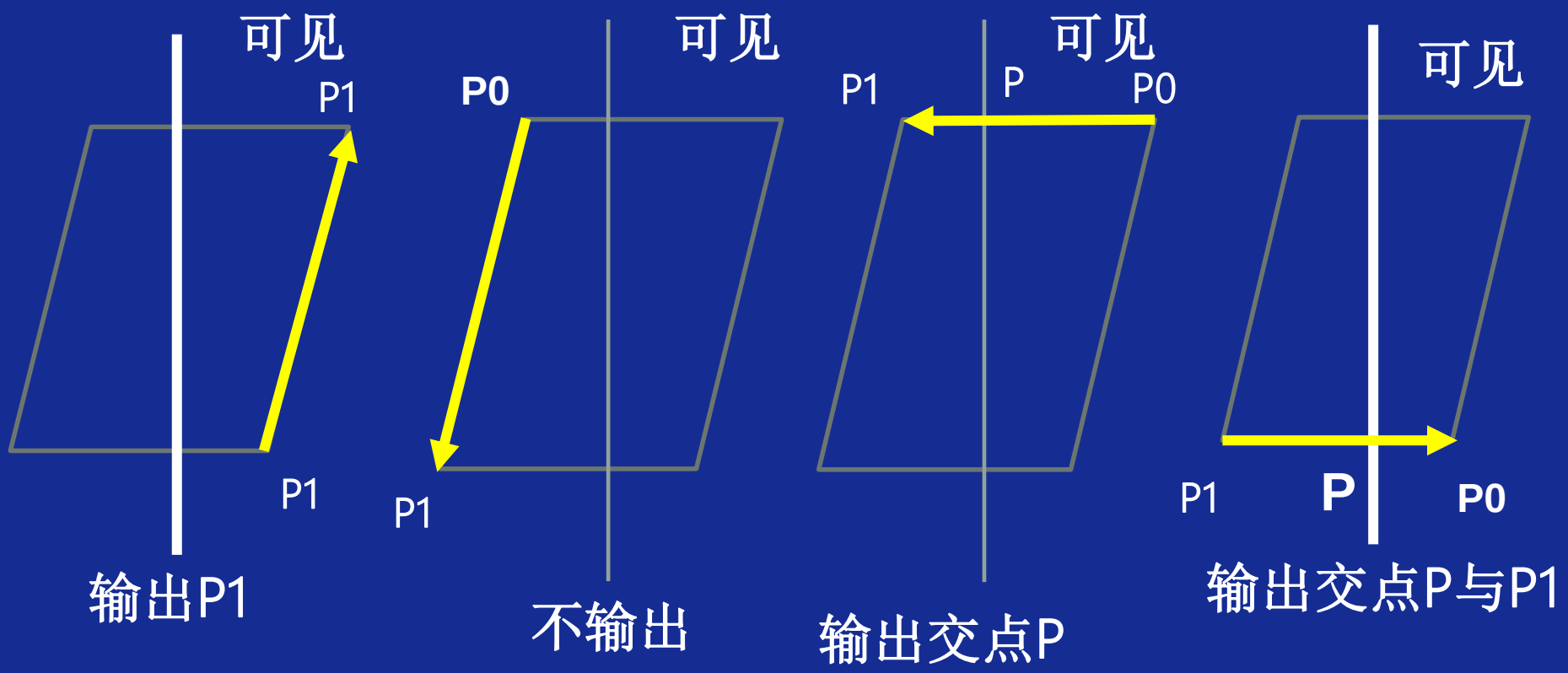
一次用窗口的一条边裁剪多边形(上、下、左、右边界)。

考虑窗口的一条边以及延长线构成的裁剪线该线把平面分成两个部分:可见一侧；不可见一侧。

多边形各条边的两端点P0 （起点）、 P1（终点）。它们与裁剪线的位置关系只有四种:

- 1) P0 、 P1均在可见侧，输出终点P1(1个顶点)；
- 2) P0 、 P1均在不可见侧，没有输出(0个顶点)；
- 3) P0可见， P1不可见，则输出线段P0 P1与裁剪边界的交点P(1个顶点)。
- 4) P1可见， P0不可见，则输出线段P0 P1与裁剪边界的交点P和终点P1(2个顶点)；

多边形裁剪



5.8 多边形裁剪(重点)

对于情况（1）仅输出顶点P1；

情况（2）输出0个顶点；

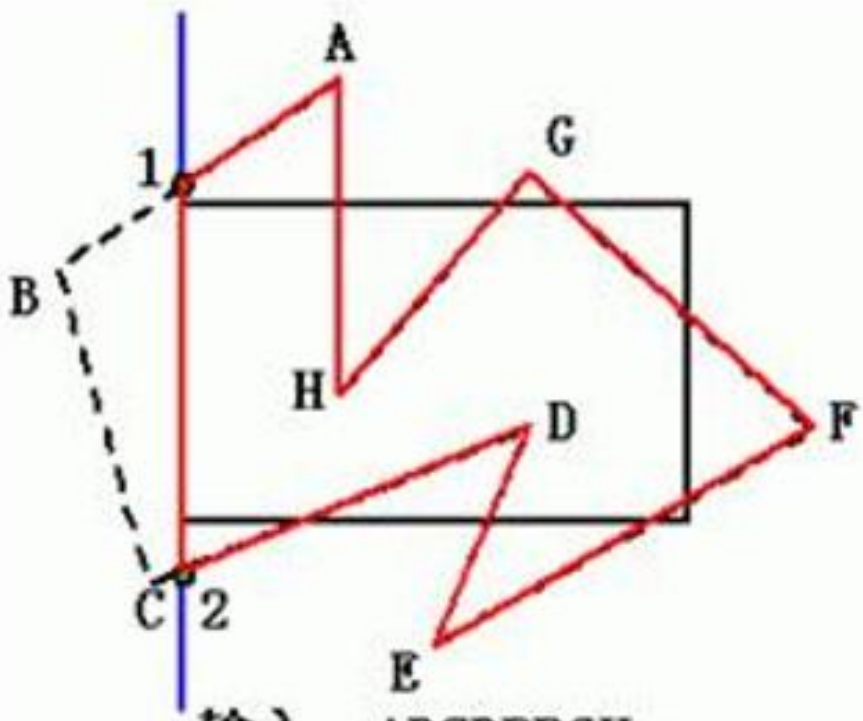
情况（3）输出线段P0P1与裁剪线的交点P；

情况（4）输出线段P0P1与裁剪线的交点P和终点P1

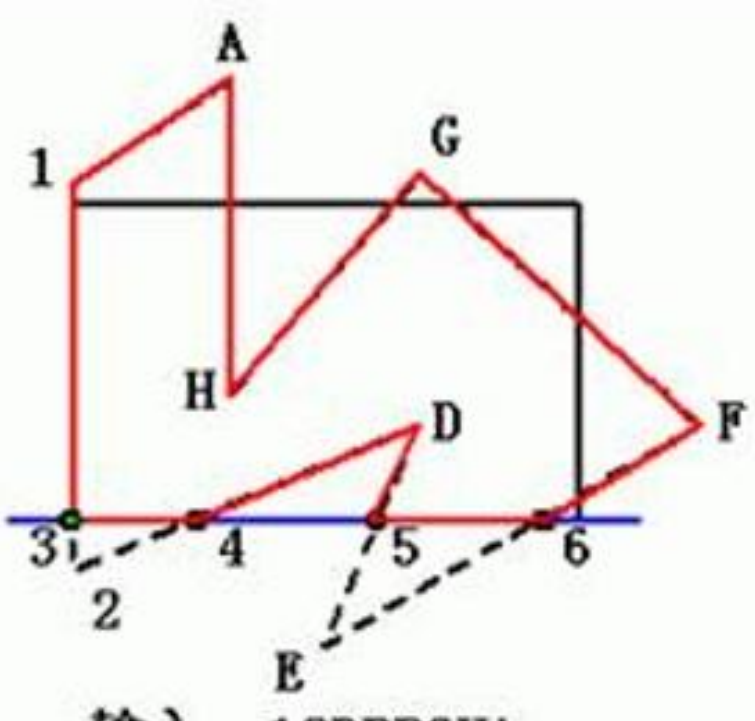
上述算法仅用一条裁剪边对多边形进行裁剪，得到一个顶点序列作为下一条裁剪边处理过程的输入。

对于每一条裁剪边，算法框图同上，只是判断点在窗口哪一侧以及求线段SP与裁剪边的交点算法应随之改变。

多边形裁剪

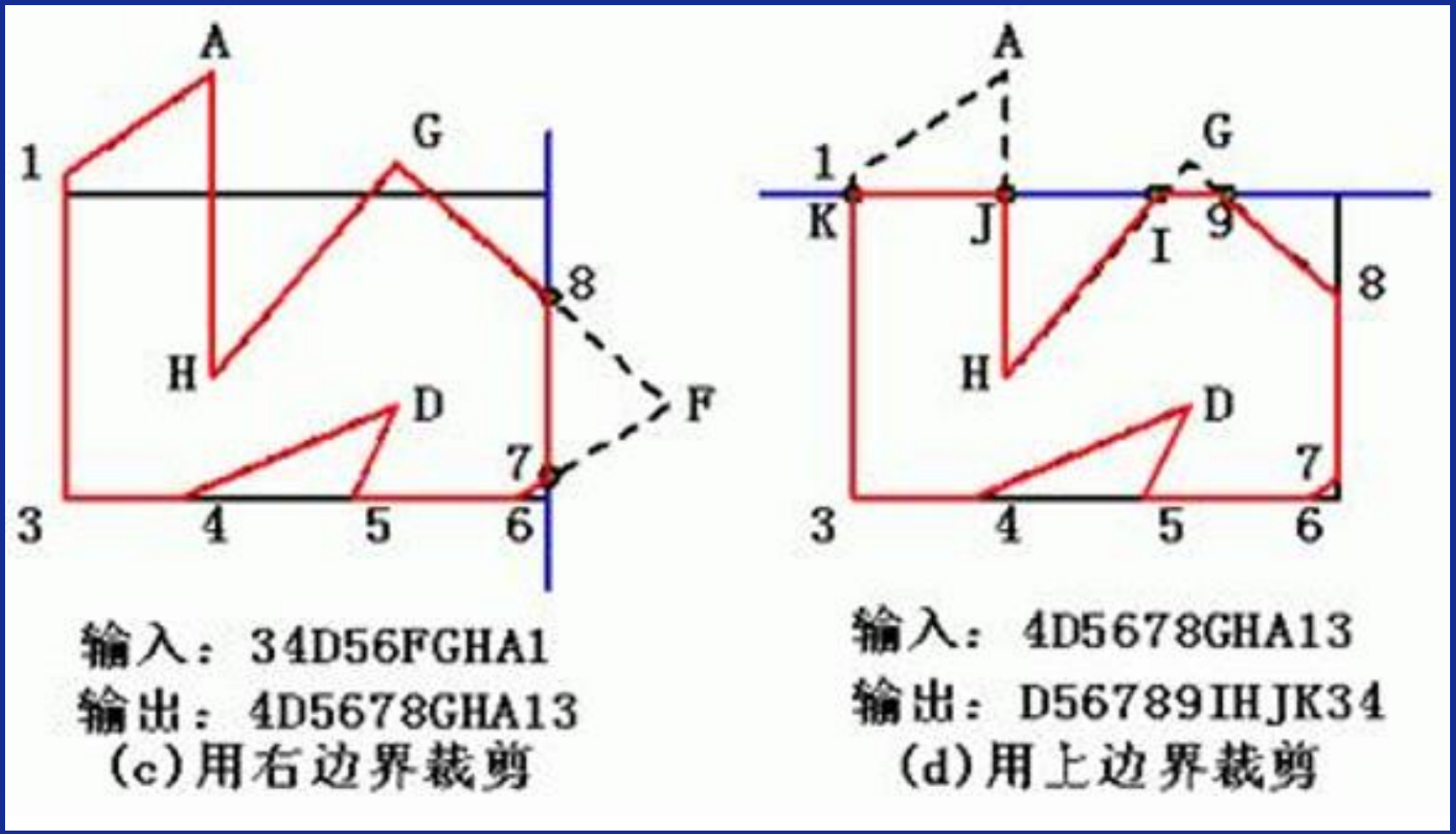


输入：ABCDEFGH
输出：12DEFGHA
(a)用左边界裁剪

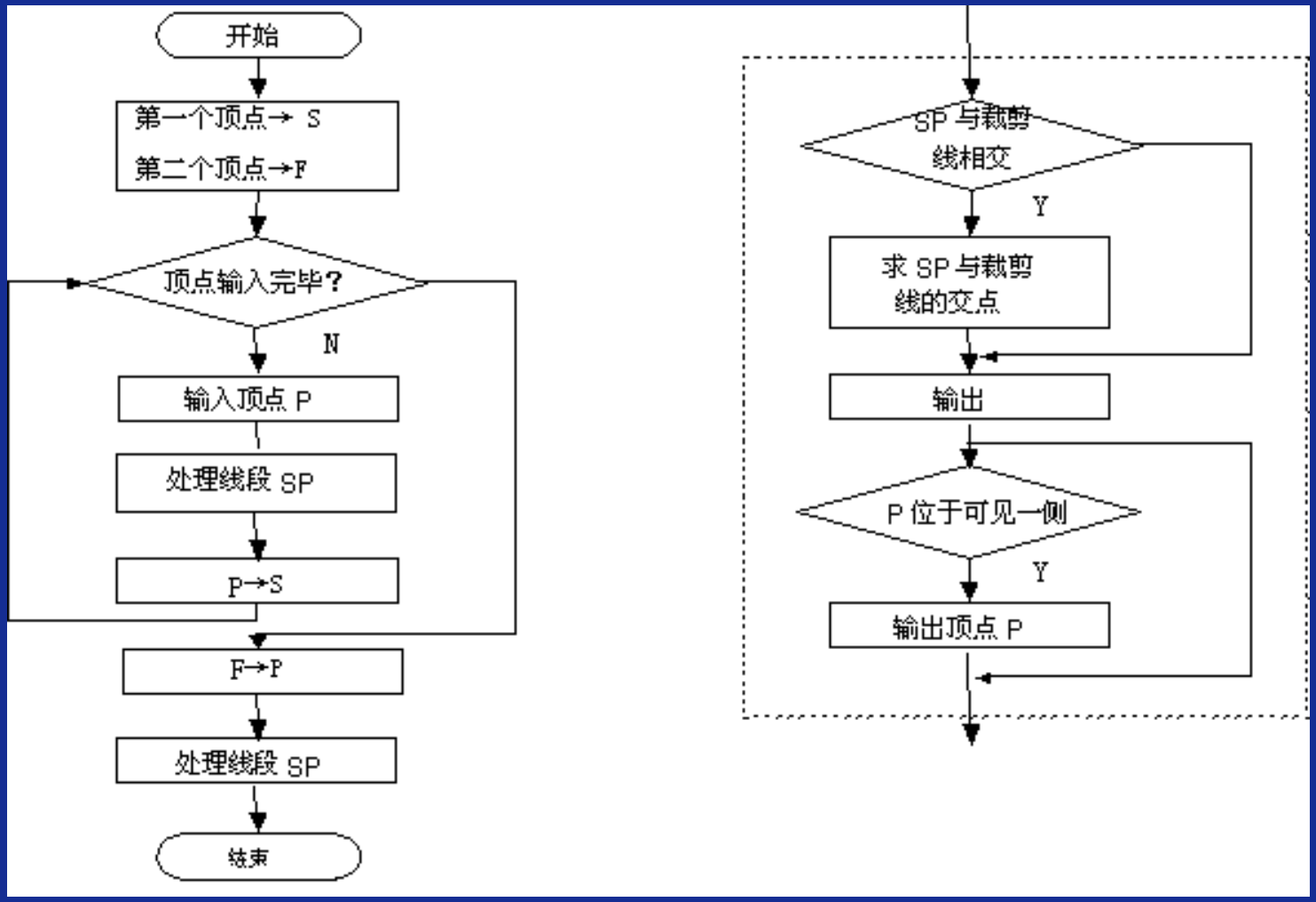


输入：12DEFGHA
输出：34D56FGHA1
(b)用下边界裁剪

多边形裁剪


A
G
1
8
H
D
F
7
3
4
5
6
输入：34D56FGHA1
输出：4D5678GHA13
(c)用右边界裁剪
A
G
1
K
J
I
9
8
H
D
7
3
4
5
6
输入：4D5678GHA13
输出：D56789IHJK34
(d)用上边界裁剪

仅用一条裁剪边逐次裁剪多边形的算法框图



字符裁剪

- 当字符和文本部分在窗口内，部分在窗口外时，就出现字符裁剪问题。字符串裁剪可按三个精度来进行，包括：

  串精度:将包围字串的外接矩形对窗口作裁剪

  字符精度:将包围字的外接矩形对窗口作裁剪

  笔画₩像素精度:将笔划分解成直线段对窗口作裁剪

待裁剪字符串　　串精度裁剪　　字符精度剪裁　　像素精度剪裁

5.9 本章小结

- 二维变换需要掌握基本几何变换矩阵。
- 二维图形基本几何变换的平移、比例、旋转、反射和错切是仿射变换的特例，反过来，任何仿射变换

$$\begin{cases} x' = ax + by + m \\ y' = cx + dy + n \end{cases}$$

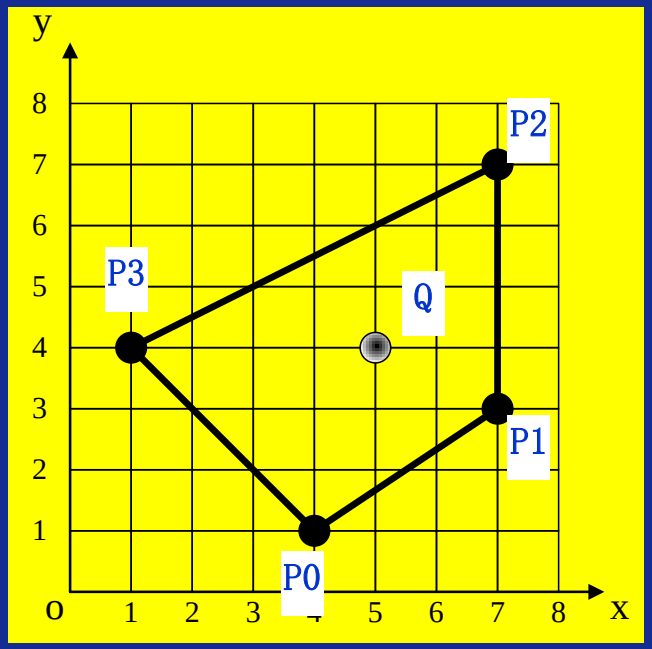
- 总可以表示为这五种变换的组合。

5.9 本章小结

- 二维裁剪属于二维观察的内容，窗口建立在观察坐标系、视区建立在屏幕坐标系。窗口和视区大小一致
- 本章给出了三种直线裁剪算法：
  - 其中Cohen-Sutherland算法是最着名的直线裁剪算法，创新性地提出了直线端点的编码规则，但这种算法需要计算直线和窗口的交点；
  - 中点分割裁剪算法避免了求解直线和窗口边界的交点只需计算直线中点坐标就可以完成直线的裁剪。
  - 梁友栋 - Barsky算法是效率最高的直线裁剪算法，通过参数t的计算，把二维裁剪问题转化成一维裁剪问题，直线的裁剪转化为求解一组不等式的问题。

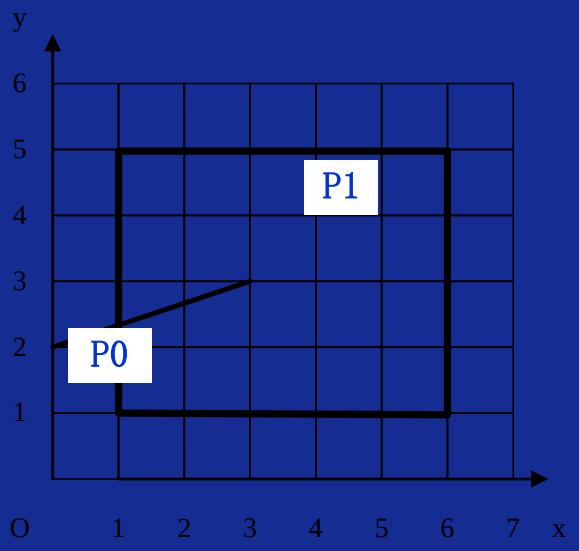
习题5

1、如图5-28所示，求P0（4，1）、P1（7，3）、P2（7，7）、P3（1，4）构成的四边形绕 Q(5，4)逆时针旋转45°的变换矩阵和变换后图形的顶点坐标。



习题5

4、用Cohen-Sutherland算法裁剪线段P0(0，2)，P1(3，3)，裁剪窗口为w_{xl}=1，w_{xr}=6，w_{yb}=1，w_{yt}=5，如图5-31所示。要求写出：（1）窗口边界划分的9个区间的编码原则。（2）线段端点的编码。（3）裁剪的主要步骤。（4）裁剪后窗口内直线段的端点坐标。



习题5

5.窗视变换公式也可以使用窗口和视区的相似原理进行推导，但要求点P（x_w，y_w)在窗口中的相对位置等于点P′（ x_v，y_v）在视区中的相对位置，请推导以下的窗视变换公式：

$$\begin{cases} x_v = a \cdot x_w + b \\ y_v = c \cdot y_w + d \end{cases}$$

变换系数为：

$$\begin{cases} a = S_x = \dfrac{vxr - vxl}{wxr - wxl} \\ b = vxl - wxl \cdot a \\ c = S_y = \dfrac{vyt - vyb}{wyt - wyb} \\ d = vyb - wyb \cdot c \end{cases}$$

2.5裁剪